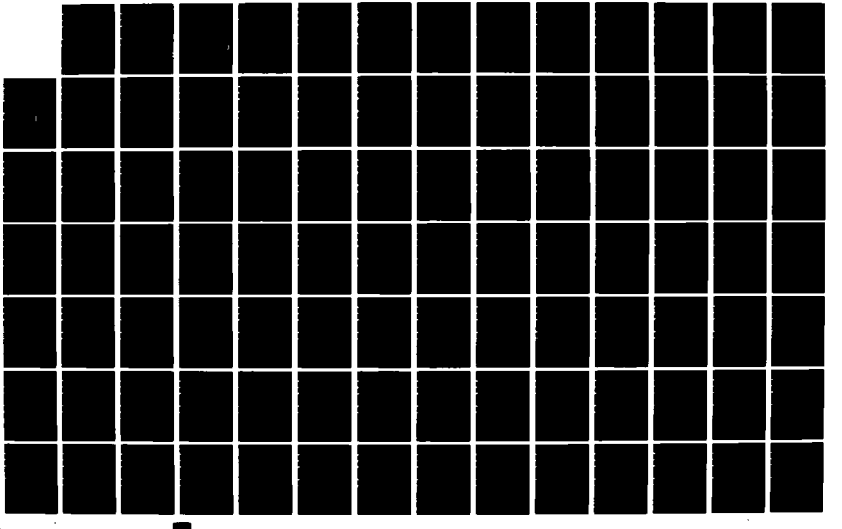
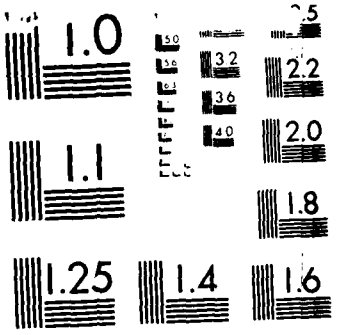
AD-A164 035 REAL-TIME FLIGHT TEST PCM DATA ACQUISITION MONITOR(U) 1/3
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
OF ENGINEERING J R CROASDALE SEP 85 AFIT/GE/ENG/85S-1
UNCLASSIFIED F/G 17/2 NL





REAL-TIME FLIGHT TEST PCM DATA ACQUISITION MONITOR

THESIS

John R. Croasdale
Lieutenant Colonel, USAF

AFIT/GE/ENG/85S-1

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GE/ENG/85S-1

REAL-TIME FLIGHT TEST PCM

DATA ACQUISITION MONITOR

THESIS

John R. Croasdale
Lieutenant Colonel, USAF

AFIT/GE/ENG/85S-1

Approved for public release; distribution unlimited

AFIT/GE/ENG/85S-1

REAL-TIME FLIGHT TEST PCM DATA ACQUISITION MONITOR

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

John R. Croasdale, B.S.

Lieutenant Colonel, USAF

September 1985

Preface

The purpose of this undertaking was to provide the 4950th Test Wing with an inexpensive way to monitor data being recorded during a test flight and prevent the circumstance where faulty instrumentation equipment could waste valuable flight time if undetected. Although only a prototype, the system developed will provide the baseline upon which further development of the PCM monitor into a flight qualified unit will proceed.

This project is dedicated to my wife for all the inconveniences and the hours she had to watch the kids when I was working on it.

I would like to thank the Instrumentation Engineering Branch of the 4950TEST WING's Directorate of Flight Test Engineering for the help provided in defining the requirements and final test of the system. Without their support, the PCM MONITOR could not have been produced.

I would also like to thank my advisor, Captain Dave King, for the help he gave me in the preparation of this thesis.

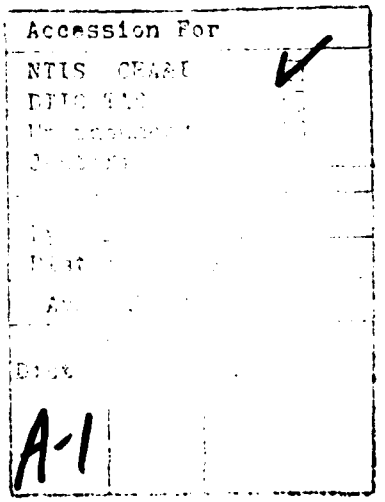


TABLE OF CONTENTS

List of Figures

Figures

List of Tables

Abstract

A computer based system utilizing an inexpensive off-the-shelf personal computer and original interface design centered around a 68000 microprocessor for real-time monitoring of a time division multiplexed pulse code modulated (TDM/PCM) data stream was designed and constructed. This system is a prototype of a low-cost, portable PCM data acquisistion monitor intended for use in flight test programs by the 4950th Test Wing at Wright-Patterson AFB, Oh. It will accept a single Armed Forces Instrumentation Standard NRZ or split-phase (Manchester) baseband data stream at rates up to 100 KBPS, display selected data words in graphical or numerical format, and alarm the user when data exceeds certain limits. It will provide a real-time verification that the data being generated and recorded during a test is of acceptable quality, allowing the option of continuation of the test, or termination. The system is capable of automatically determining the data rate and signaling format and synchronizing itself with the incoming signal.

I. Introduction

Purpose

This thesis describes the design of a microprocessor controlled Pulse Code Modulation (PCM) display system (monitor) which can be used to present selected data inbedded within a PCM stream. The unit will be referred to as the PCM monitor throughout this report. Its intended use is to provide a flight test program with a real-time display capability of test parameters which could be used by a test director to evaluate the progress of his/her test during a flight. Current methods involve recording an entire PCM data stream on an analog tape recorder and reading it back later on the ground at a reduced rate using laboratory equipment. This method provides the accuracy required in a test report but in cases where malfunctioning instrumentation equipment supplies erroneous data or no data at all, an entire test flight can be wasted. An inexpensive PCM monitor could be used to verify the integrity of the data while it is being recorded and alert the test director of problems before valuable flight time costing as much as $5000 per flight hour is lost. If it is determined that the quality of the PCM data is unacceptable, the test flight profile could be altered to improve performance, or terminated.

There are a few commercial systems costing around $80,000 that could provide such a monitoring function. The procurement costs for the approximately 18 units required by the 4950th Test Wing, however, would be prohibitive. This thesis describes one alternative which could be used to supply this urgently needed function at an estimated cost of around $5,000 per unit.

Scope

The scope of this project was limited to to the design and construction of a laboratory based PCM monitor tailored to the types of PCM commonly used in the 4950th Test Wing. These PCM types are NRZ-L and Bi-Phase-L at rates of around 50Khz. The design and construction of a flight qualified unit will be left as a follow-on effort by people in the Wing qualified in that area.

The effort resulted in a PCM monitor system which consists of two parts: the PCM processor and the PCM display processor. The system is portrayed in figure 1. The software for the PCM processor was written to provide several features such as automatic bit rate calculations and automatic PCM type selection. The display processor software was written to provide a user-friendly environment to allow access to the required data without extensive documentation or research. The software design was limited to a demonstration quality showing the feasibility and flexibility of the PCM monitor system.

Background

The 4950th Test Wing provides for flight testing for various systems in different stages of development. It performs development and operational testing of avionics systems, aero evaluations of aircraft Class II modifications, and system test support of some fairly large programs such as the Space Shuttle and MX missile using the Advanced Range Instrumentation Aircraft (ARIA). In each phase of these functions, data collection and recording is paramount to the test effort. Without such collection and recording, all data would have to

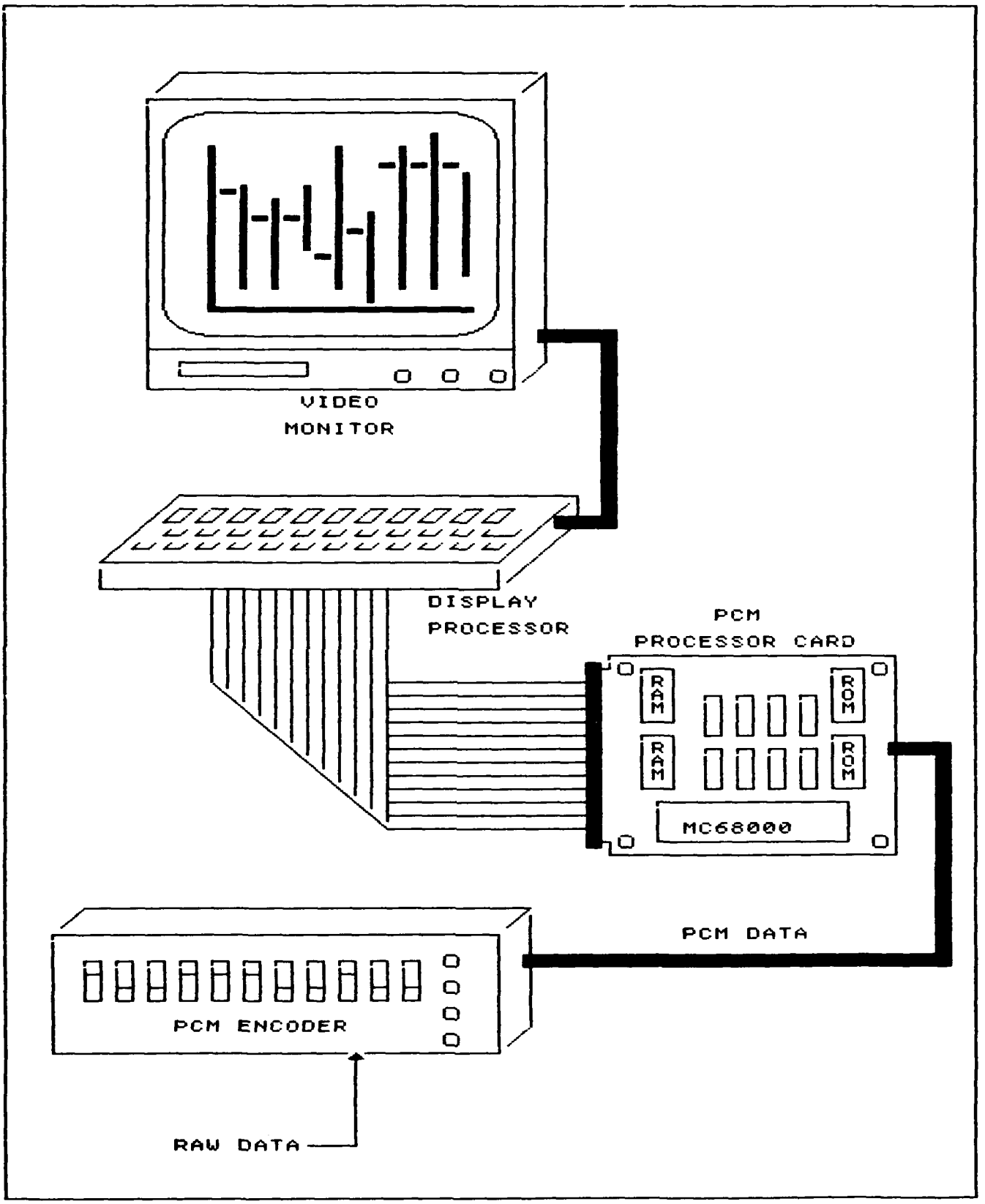


Figure 1. PCM Monitor Pictorial

be analyzed during the flight where it would be extremely difficult if not impossible. It is therefore imperative to record many different data sources in the most expedient way possible. A common way to do this is with the use of Time Division Multiplexing (TDM)/ Pulse Code Modulation (PCM) techniques.

A TDM/PCM signal is a serial digital data stream. It consists of a sequence of frames. Each frame in turn is composed of a specified number of digitized samples of data called data words, each a specified number of bits in length. Within the frame, several of the data words contain a unique series of bits which are used to establish synchronization between the transmitting and receiving equipment. PCM allows a large number of slowly varying types of data to be packed into a single electronic data stream which can be recorded easily. Many data items or parameters sampled during a flight test require very slow sample rates on the order ten times a second or less. Such parameters include control surface position, altitude, airspeed, engine temperature etc. Recording items like these independently would be very wasteful; therefore, PCM is used.

Electrically, the data is encoded as different voltage potentials between two wires. Normally one of the lines is at ground potential and the other is either at a higher positive voltage representing a digital "1", or grounded represented a digital "Ø". By varying the voltage with respect to time, a serial data word is transmitted to a receiving unit as a series of Ø's and 1's. Groups of these bits, as they are called, form words of data while series of words form frames.

In PCM, the signal train consists of one or more frames, each

containing one or more synchronization words and one or more data words. The synchronization words must be of sufficient composition and length to be distinguished from the data words. Normally two to three synchronization words provide a bit pattern unique enough for all practical purposes and provide high reliability. The receiving unit reads in the data bits looking for the synchronization words. When it detects a match, it is considered to be synchronized and begins reading in the data correctly. Without proper synchronization, reading groups of bits into words would be meaningless as each bit has a value depending on its relative position in the word.

Normally each data word has a specific meaning. Perhaps the user wants to record the deflection angle of all control surfaces in the aircraft with respect to time during a test flight. This type of test is commonly done in airworthiness evaluations of new aircraft. Each control surface such as ailerons, rudder, elevator, flaps, trim tabs etc., would have a device known as a transducer which outputs a signal which relates to the deflection angle of the surface it is measuring. This signal is sampled, quantized, and encoded, then combined with all other control surface words into a PCM data stream consisting of 1's and Ø's. Common requirements are that all words must be the same length, the number of words must be the same for each frame, and the words must be in the same order within the frame. The PCM system which accepts these words and forms the stream is known as a PCM encoder. It samples the data inputs, inserts them into a predefined order, attaches synchronization words, then transmits to one of several types of equipment. Once the last data word is sent, the frame is complete and

the whole process starts over. The method of building the PCM train is commonly known commutation. Likewise, the method of separating the data stream into distinct data words is called decommutation.

There is a method called subcommutation which actually changes the meaning of a specific data word according to the frame number in a very precise order. For example, data word one in frame one might represent the left aileron position. In frame two it might represent the right aileron position. In frame three it might revert back to the left aileron position etc. If this type of commutation is used, some sort of frame identification must be incorporated in one or several of the data words. The decoder at the other end must use this identification to decommutate PCM stream. Subcommutation is not used in the 4950th Test Wing and is not requirement for the PCM monitor.

PCM as a most general type of serial transmission of data. Unlike the RS-232 standard which allows computers to send asynchronous data to terminals, modems, and printers etc, PCM does not have start or stop bits, nor a dead time between packets or words of information. PCM is always active which makes it considerably more complicated to decode. PCM also comes in a wide variety of formats. Nine commonly used signalling formats are explained below in figure 2.

Only two of the formats mentioned are utilized in the 4950th Test Wing, NRZ-L and Bi-Phase-L and are described in detail in figure 4.

The primary advantage of NRZ-L is the its high bandwidth efficiency of 2 bits/Hz. For example, the bit sequence of "1,0,1,0" would represent two cycles of a square wave but 4 bits of information. The frequency for Bi-Phase-L, on the other hand, would have to be twice

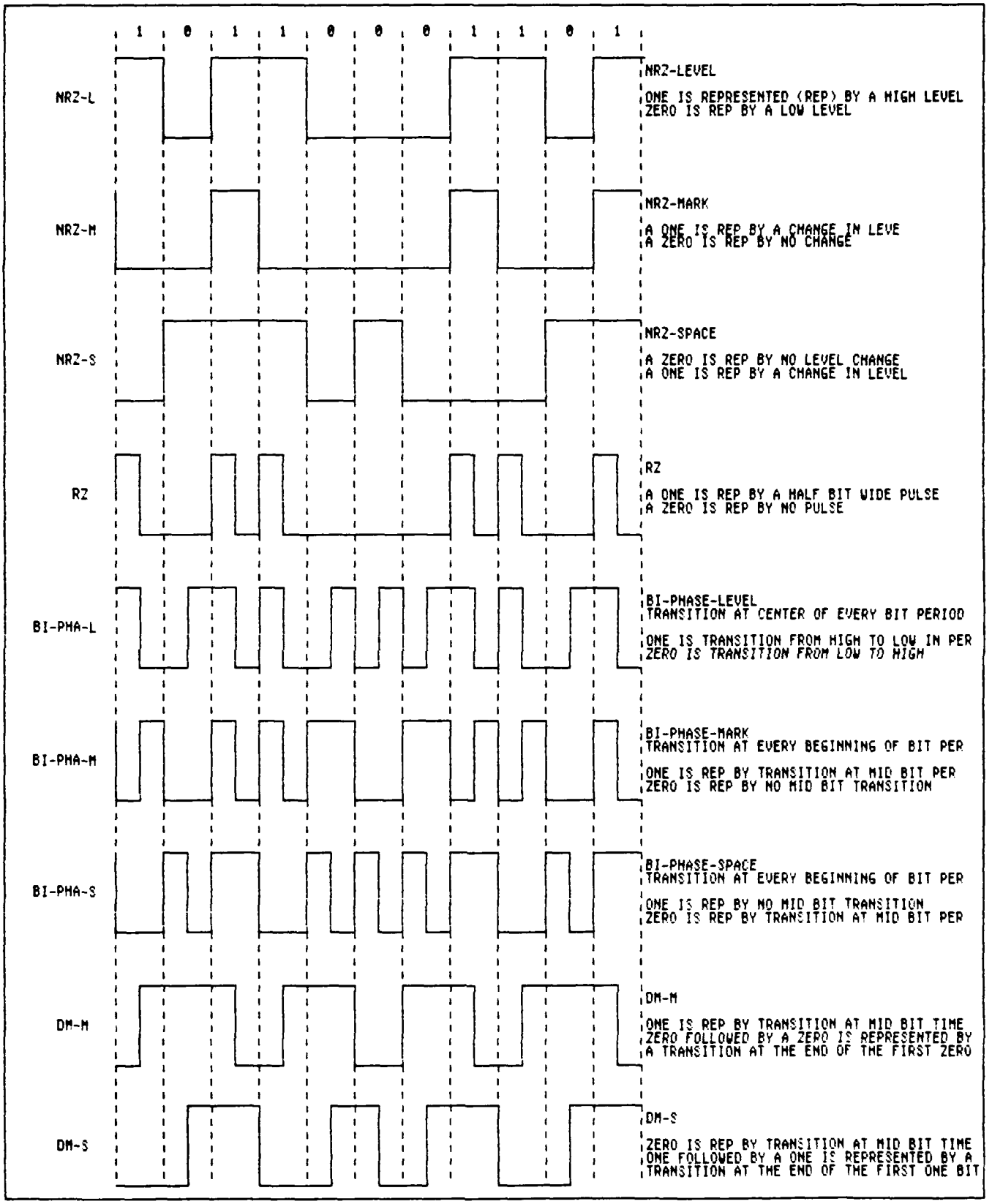


Figure 2. Standard PCM Formats

as large to achieve the same bit rate since it requires a transition during each bit time.

The disadvantage of NRZ-L is that it requires a recording system which has a DC response and thus doesn't lend itself to recording on magnetic tape directly. Since tape recorders have a low end to their frequency response curve, they can't record direct current. Therefore recording a long series of 1's or Ø's would cause data dropouts on the tape and a subsequent loss of data unless expensive frequency modulation (FM) techniques are used. In addition to the recording problems, a long series of 1's or Ø's would cause the synchronization circuitry at the receive end to lose frame synchronization because a non-varying signal could be interpreted as no signal at all.

Bi-Phase-L, on the other hand, is readily adapted to both recording and synchronization. It is the standard PCM type used when the data rate is low and needs to be recorded. In the 4950th Test Wing, recorders exist which have the capability of recording frequencies up to 1MHZ which relates to 1 megabits per second of Bi-Phase data or 2 megabits per second of NRZ. The majority of data is recorded, however, at rates of 50 Khz and below which allows long recording periods consisting of up to an hour or more on a single tape. Recording at 2 megabits of NRZ-L would allow only 12.5 minutes of continuous recording.

As mentioned above, PCM is one of the most commonly used methods of collecting the vast amounts of data needed on any particular test program. The PCM monitor will provide a valuable window into the test data while it is gathered to allow the test director to make decisions

about how to alter the test to maximize its efficiency. It is the purpose of this thesis to provide an economical way to accomplish this monitoring task.

Requirements

The requirements for the PCM monitor system are based on current types of PCM used in the 4950th Test Wing. These requirements are summarized below. The PCM monitor must:

1. Fit in a standard rack, 19" by 4.5" by 17".
2. Handle data rates from 1000hz to 100khz.
3. Decode NRZ-L and Bi-phase-L.
4. Decode word length of 8 to 16 bits.
5. Accept frame sync words of 16 to 48 bits in length.
6. Have a display rate of at least 10 data words per second.
7. Allow the user to display data in engineering units.
8. Allow the user to display data in bar graph format.
9. Be expandable.

System Description

In order to meet the above requirements, a fast and flexible system is needed. Modern day microprocessors can provide this capability so the PCM monitor was designed around them. The monitor is broken down into two modules: the PCM processor and the display processor.

The PCM processor's function is to receive, condition, synchronize, decommutate, and store PCM data into a memory buffer common to itself, and the display processor. To perform these functions, an MC68000 16 bit microprocessor is used because of its speed, asynchronous capabilities, and ease of interface. The software is written in assembly language because of the speed requirements. It

will not need to be altered under normal conditions and is programmed into Read Only Memory (ROM).

The display processor reads in the decommutated data and provides user interaction as to how it is to be displayed. It is built around a Radio Shack Color Computer II microcomputer using an MC6809 processor to provide commonalty between it and other systems under development in the 4950th Test Wing. It also has a very powerful BASIC language, has a large easily readable display format, is small, light, reliable, inexpensive, and available off-the-shelf. The majority of the display software is written in BASIC to provide for the ease of modification. Portions of the display software which are awkward to handle in BASIC and slow are coded in assembly language.

This multiprocessing approach partitions the tasks into two independent sections to allow for maximum speed and flexibility. Since the tasks are completely independent except for synchronization explained later in the display processor software section, the PCM processor can be used in any other system with only minimum re-work of the interface.

II. PCM processor

Hardware Overview

The PCM monitor is partitioned into two sections, the PCM processor, and the display processor. Because the display processor uses an off-the-shelf Color Computer II from Radio Shack, its design was primarily a software development effort (see chapter III). The PCM processor, on the other hand, is hardware intensive and is the primary thrust of this project. Its primary purpose is to perform the bit synchronization and decommutation for a standard NRZ-L and/or bi-phase-L PCM signals. To do this function, the PCM processor consists of nine sections: 1)signal conditioner, 2)system clock, 3)system controller, 4)bit timer, 5)shift generator, 6)shift selector, 7)word processor, 8)random access memory, and 9)I/O port ram. Each of these sections will be explained in this chapter. Figure 3 shows the overall block diagram and pictorial of the system.

The PCM processor receives the PCM data, reduces most of the noise and drift, synchronizes to it, performs a serial to parallel conversion, then places it into a dual port RAM buffer as 16 bit parallel data words which can be accessed by the display processor. There are two built-in automatic features which can be overridden if the user desires. These features provide for automatic clock generation and PCM type determination. In other words, the user does not have to know the data rate or the PCM type (NRZ-L or bi-phase-L).

In order to perform these functions, the PCM processor must first clean up the PCM data stream to eliminate noise, square up the signal,

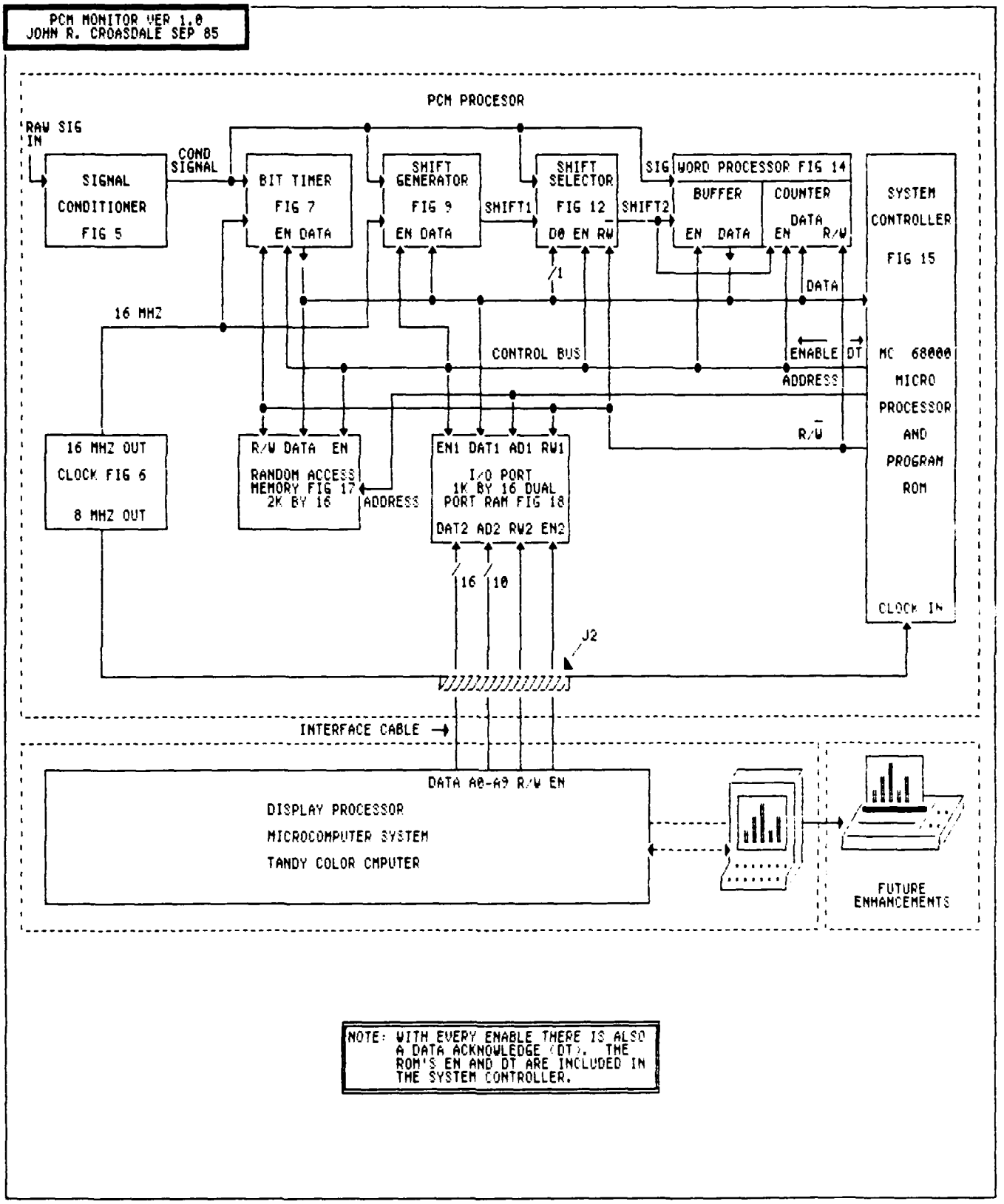


Figure 3. Block Diagram

eliminate any drift or DC offset, and convert any signal levels to TTL levels before the PCM processor will work. The signal conditioner is responsible for this task.

Once a clean TTL level signal is obtained, the PCM processor must obtain the bit rate of the incoming PCM signal. It does this in the bit timer section by counting the time interval between two bits at a high count rate, then dividing by two. In this case, a 16Mhz clock is used so each bit time is represented by a number of counts of the 16Mhz clock. Once the count is obtained, it is easy to generate a clock using another counter circuit. Since the same system clock is generating the bit rate clock from the signal, any deviation of the nominal bit rate from the actual bit rate is taken care of and will not cause any tracking error.

The shift generator uses the count obtained by the bit timer and generates shift pulses that occur at the center of the bit interval. The term "shift" is used because they will control the shift function in the word processor section described below. The level of the signal when this pulse occurs will either be at a TTL high level, or at a TTL low level. This level will represent the data bit of the transmitted message. The polarity of the signal will determine whether a high TTL level will represent a "1" or a "Ø". The display processor tells the PCM processor what the polarity is and the processor software accounts for it. A TTL high level will be assumed to be a "1" for this report. Figure 4 illustrates this process. The shift generator resynchronizes the shift pulses with each positive going transition of the input PCM signal. This insures that the pulses are continuously being

synchronized to the signal.

The shift selector modifies the shift pulses when bi-phase PCM is being transmitted to allow for proper decoding of the signal. Its output is a shift pulse that occurs at the correct time so that the input signal can be read into the word processor. For NRZ signals, the shift pulse from the shift generator is unmodified.

The word processor reads in the signal and performs the serial to parallel conversion. The heart of the word processor is a shift register which collects the incoming 1's and Ø's. The system controller can then read the shift register's output buffer 16 bits at a time and at two different rates. By reading the data after each bit is shifted in, (bit rate), the frame synchronization data word can be captured. After frame synchronization, reading the data at the word rate, (8 to 16 bits at a time), gives the controller ample time to read data word, do error checking, and perform the data transfer to the I/O Port/buffer. Once the data is in the I/O buffer, it can be read by the display processor.

The system controller is based on an MC68000 16 bit microprocessor and does the coordinating and calculations needed by the PCM processor. It reads the bit count from the bit timer, transfers the count into the shift generator, writes the number of words per frame into the word processor, calculates the PCM type, performs frame synchronization, and transfers the data from the word processor to the I/O PORT. If frame sync is not found, the user is alerted if the option is set (see software section) and the system keeps trying. The controller program is stored in EPROM. During initialization, it is moved into fast RAM

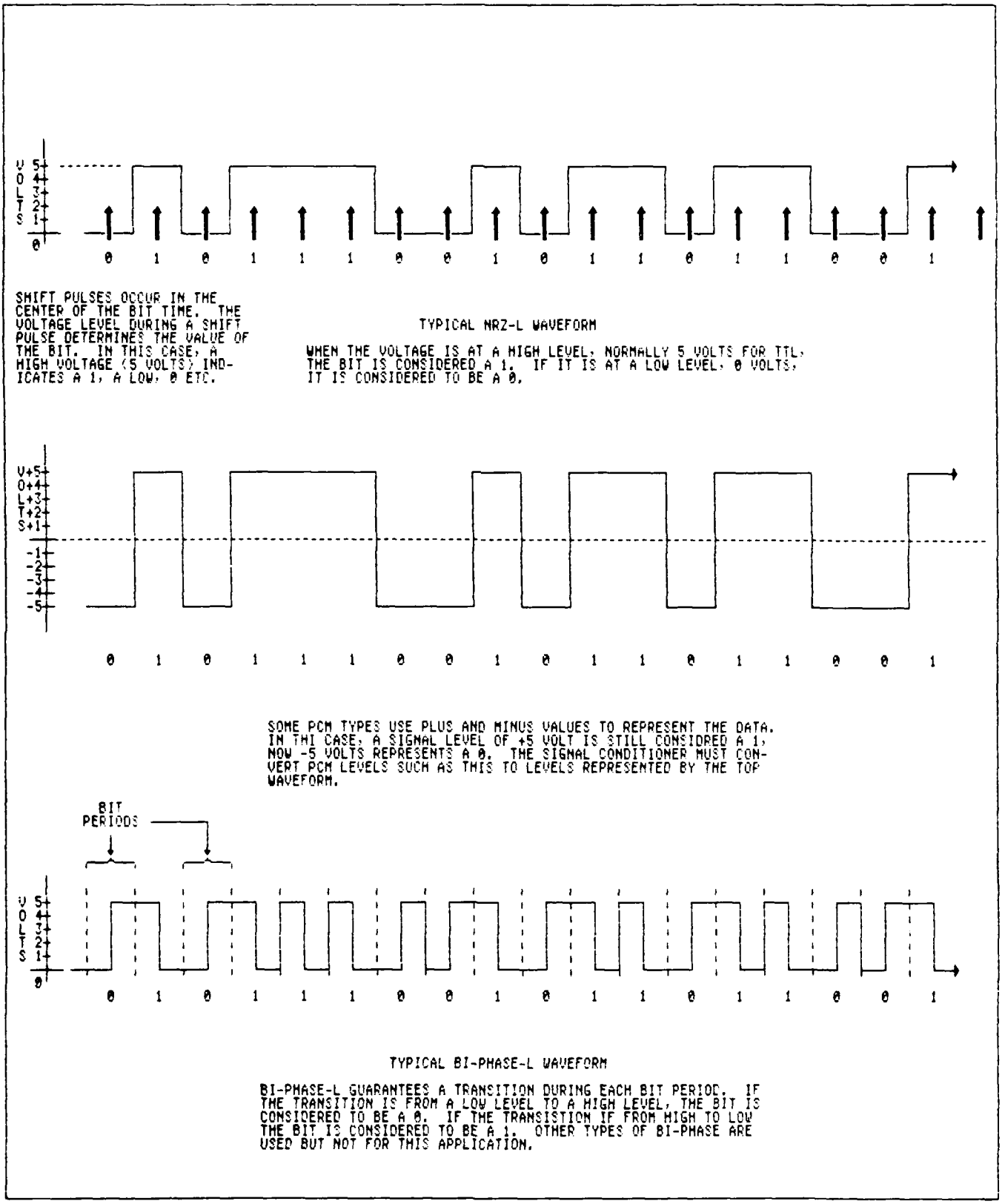


Figure 4. NRZ, Bi-Phase Explanation

where it runs at maximum speed.

The following sections explain in detail the hardware design and function of each of the above mentioned parts of the PCM monitor's PCM processor.

Signal Conditioner

The main function of the signal conditioner whose schematic shown in figure 5, is to clean up the PCM signal, and convert it to a standard Ø to 5 volt TTL signal. Since the input signal levels are not known (assumed to be at least in the Ø to 5 volt range or more), signal clipping may be necessary to avoid exceeding the input requirements of TTL chips. The diode pair keeps the signal in the Ø to 5 volt range approximately. Germanium diodes should be used to keep the junction voltage drop to around .3 volts. For example consider an input signal which uses -1Ø volts for a Ø, and +1Ø volts for a 1. When the signal is at -1Ø volts, it will forward bias D2 and keep the level at the input of 74LS14 schmitt trigger around Ø volts. When the signal is at +1Ø volts, D1 will now be forward biased and clip the signal to +5 volts. If the signal is already Ø to 5 volts, no clipping will occur and the signal will be applied directly to the schmitt trigger input. The resistor is used for current limiting to keep the circuit from loading signal source excessively if the signal is clipped.

The first schmitt trigger is used to square up the incoming signal. The trigger has very high gain so any incoming signal will either be saturated, or cutoff. This causes the output to occupy only two states, high or low. Any variation in between will be eliminated.

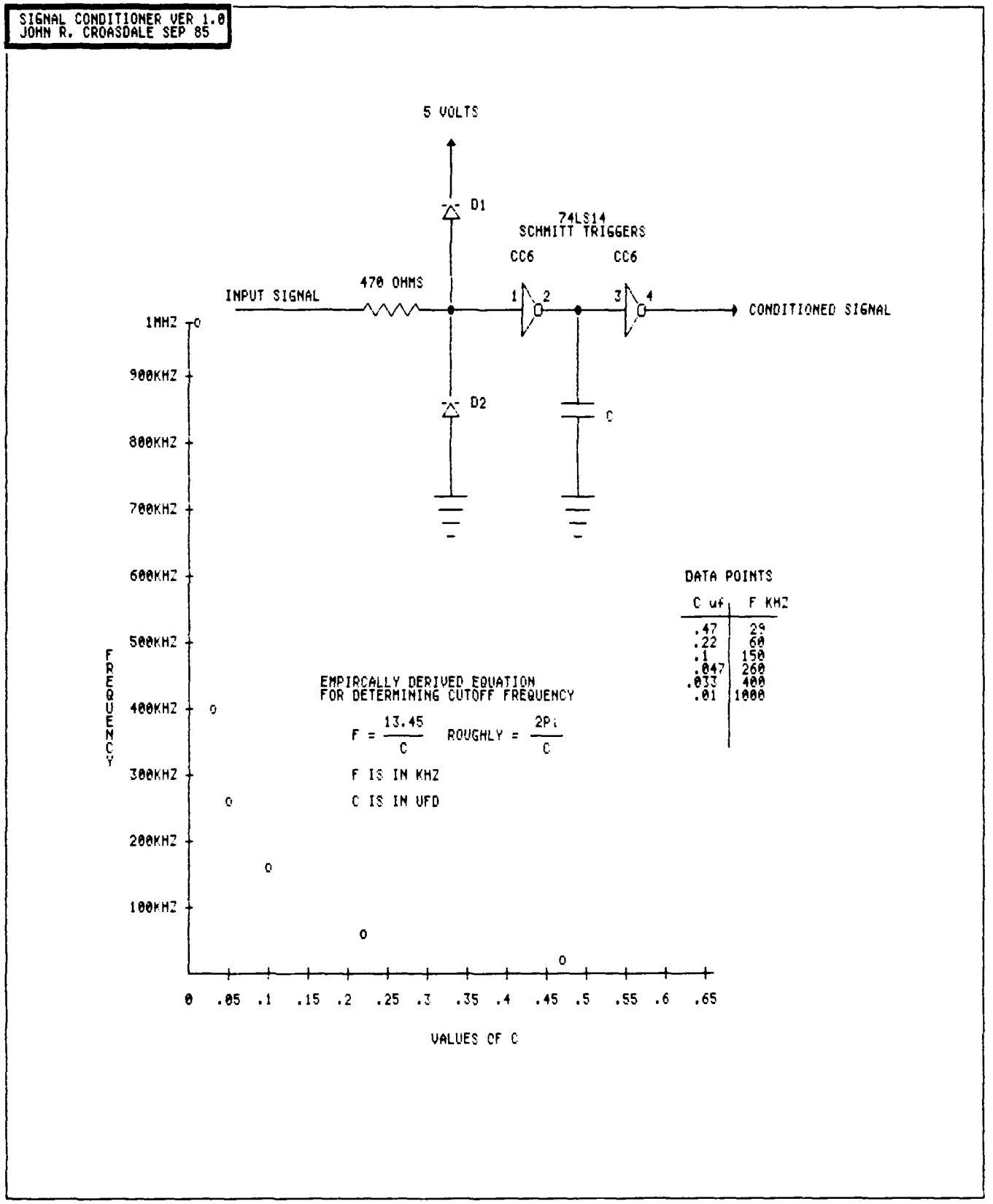


Figure 5. Signal Conditioner

The capacitor and next schmitt trigger filter any high frequency noise according to the equation shown in figure 5. The filter has a very high Q factor and a sharp cuttoff which has been limited to around 500Khz. Noise spikes shorter than 2 microseconds will be eliminated. The graph shown in figure 5 shows the cuttoff frequency in relation to the capacitor.

System Clock

The system clock provides two clock signals, a 16Mhz square wave for a counting clock for the bit timer and shift generator circuits, and an 8Mhz square wave for the microprocessor.

The circuit uses a Saronix 32 Mhz crystal controlled hybrid clock which is divided to both 16 Mhz and 8 Mhz by a 74LS93 divider IC. Exact clock frequency is not critical but the stability is; therefore, a crystal controlled device is necessary. Figure 6 shows the schematic of the system clock. The MC 68000 microprocessor does not require any elaborate time phasing or quadrature clock so a plain square wave at TTL levels is all that is necessary.

Bit Timer

The main function of the bit timer is to determine the incoming PCM bit rate automatically. In order to do this, a high speed counter senses the incoming pulses and starts counting the 16 Mhz clock pulses from the system clock when the conditioned signal goes from a low to a high (see figure 7). The bit timer toggles off when a second positive going pulse is received and makes the count available to the system

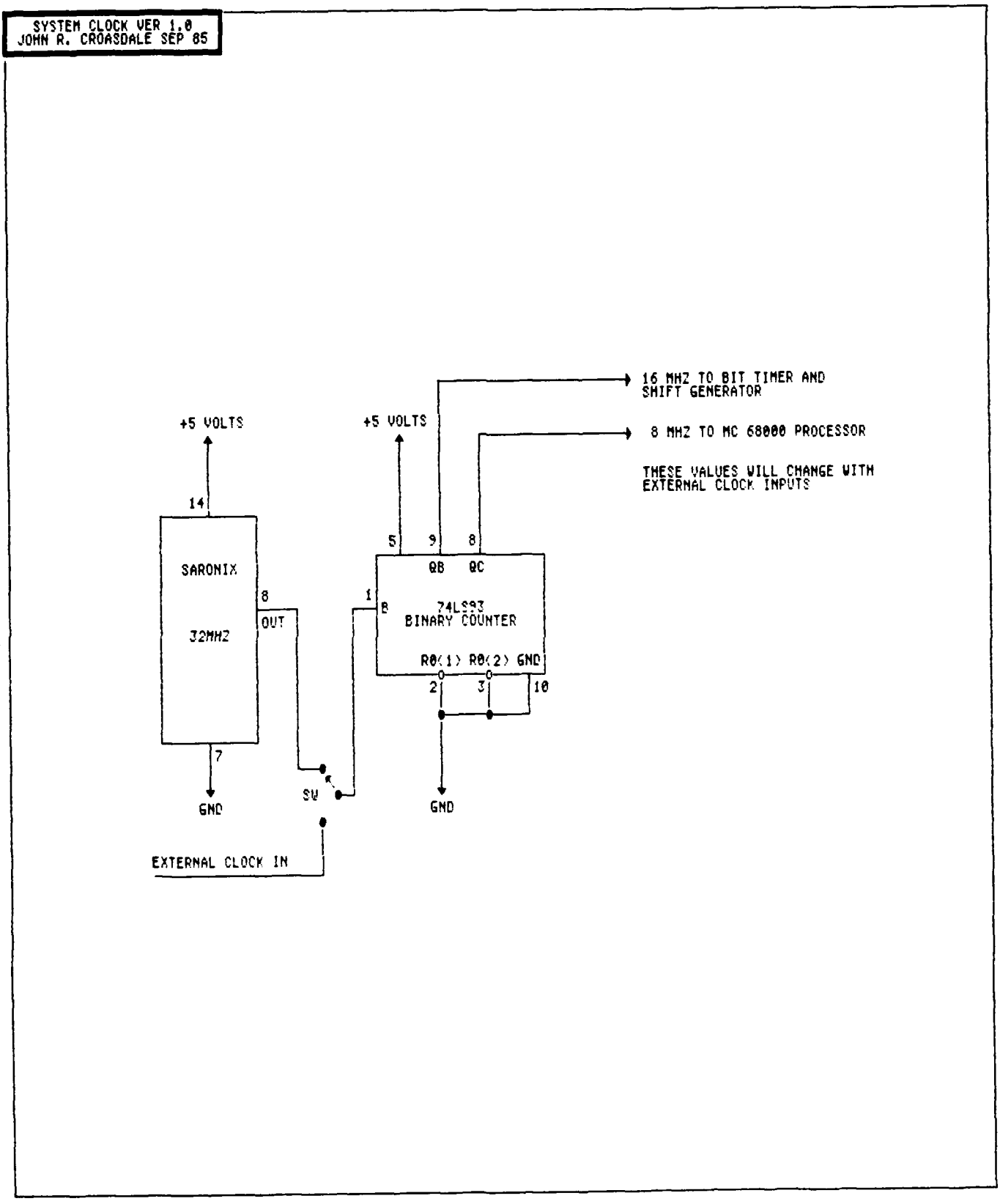


Figure 6. System Clock

controller (see section on system controller). The controller will monitor a series of count cycles and keep track of the shortest one. If a binary series of "0 1 0 1" is received anywhere in the data stream, the count will represent 2 bit times. The controller will try to synchronize using the minimum count and if it breaks lock, increment the count by one. Since the minimum count is chosen, the actual count must lie somewhere between this minimum count and some maximum value. If there is any jitter in the signal, the minimum count will probably not be sufficient for high bit rates. See PCM processor software later this chapter. For a signal of 100000 bits per second (BPS), the number of counts per bit time will be 160. The bit timer will therefore reach a count of 320 since it counts two bit periods.

To understand how the bit timer performs its function, consult figure 8. The circuit utilizes a 74LS74 flip flop to toggle the count gate (CA2) on, then off, one toggle per each positive transition of the input signal. During the on time, gate (CA2) allows the 16 Mhz clock signal to enter the binary counter network (CB1 and CB2). At the same time, the flip flop's Q output triggers a 74LS123 monostable multivibrator to generate a very short 10ns pulse to clear the counters for the upcoming next count period. When the signal toggles the flip flop a second time, the gate is closed and the count stops. Also, the Q' output enables the two 74HC374 latches which clocks in (receives) the count. The count therefore remains stable during the off time. The controller reads the count by strobing the output enable lines of the latches. When it is not reading the count, the latch outputs are

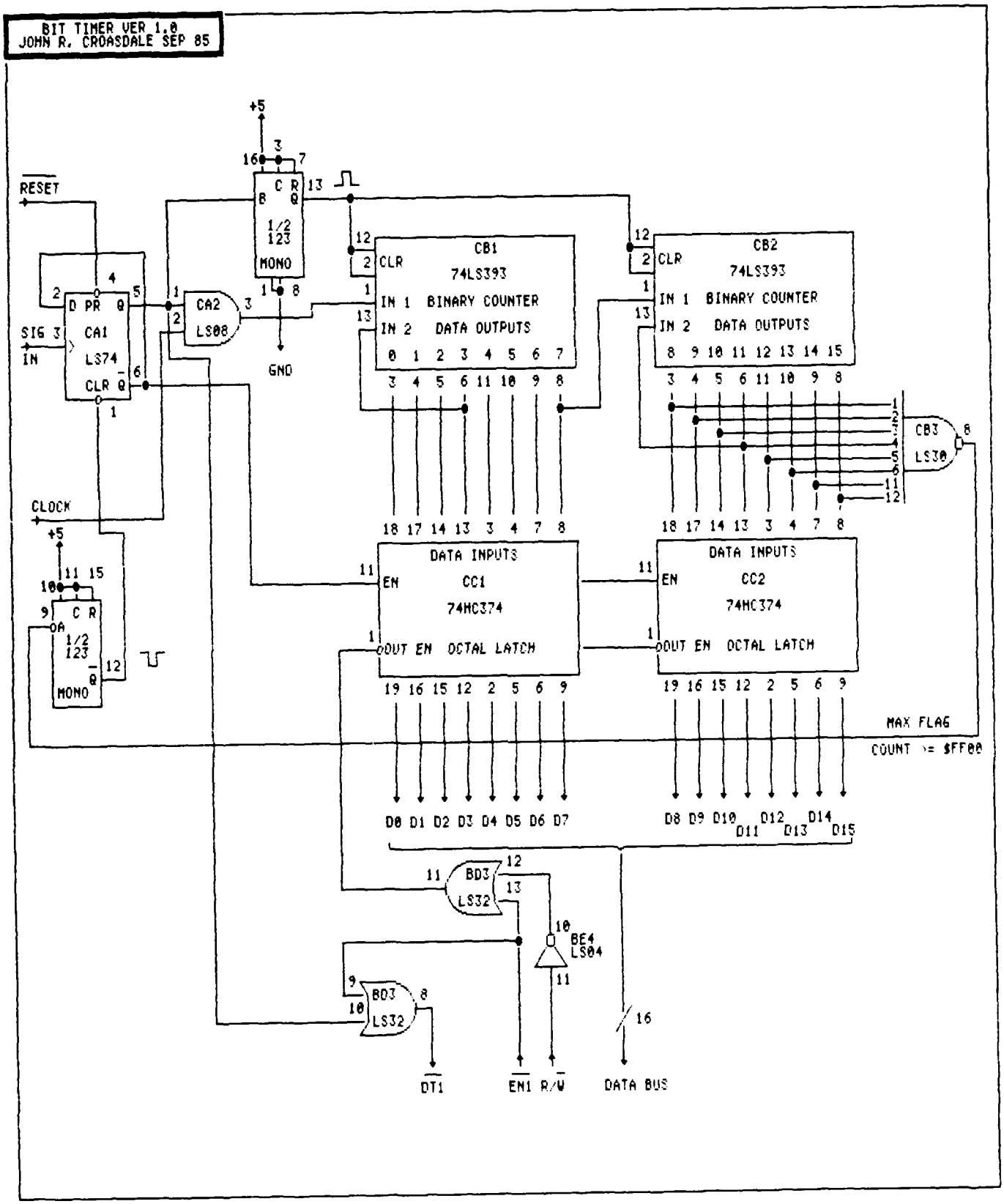


Figure 7. Bit Timer

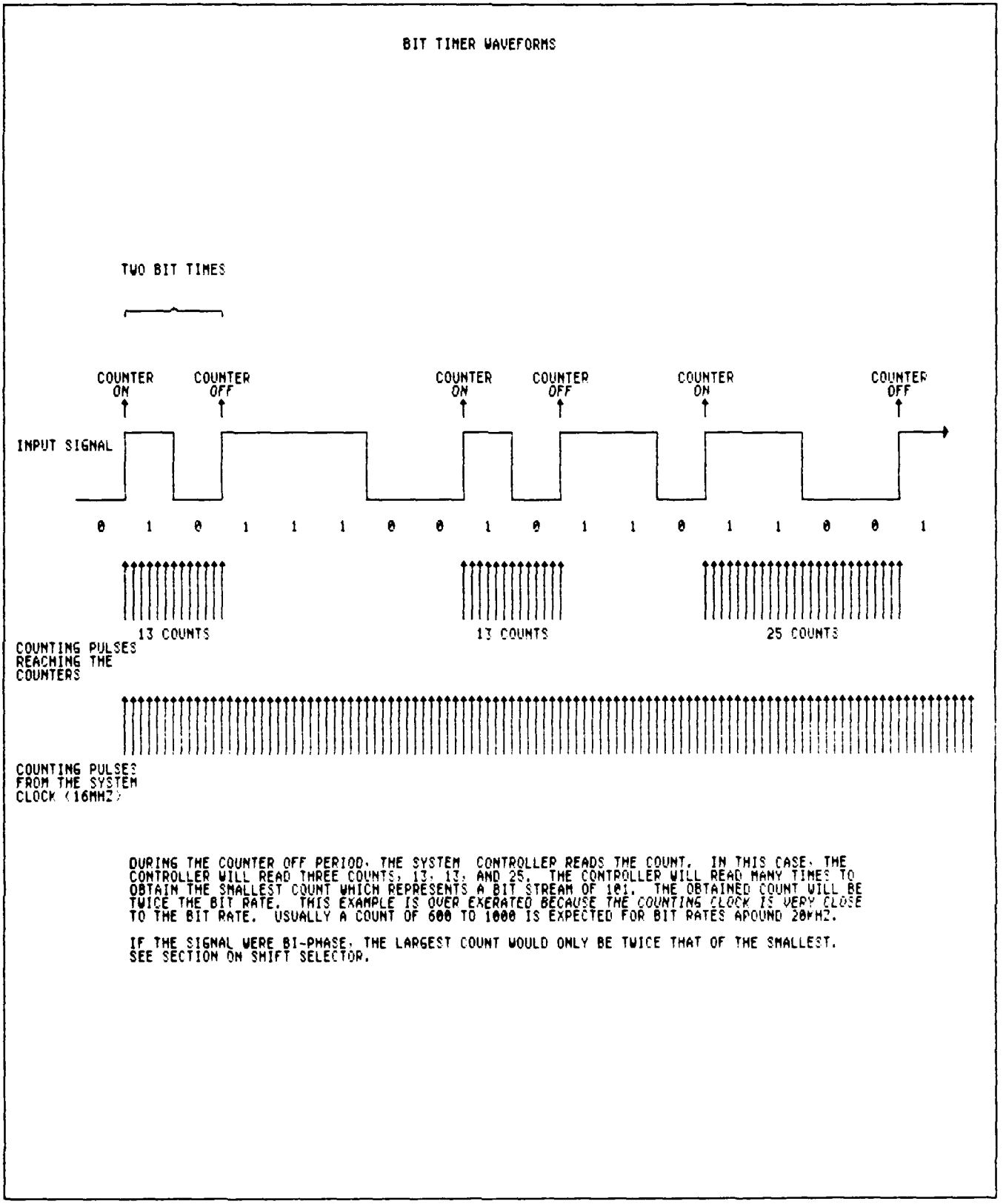


Figure 8. Bit Timer Waveforms

at a high impedance will increase to a value of 65280, or $FF00 hex. This will enable the 74LS30 NAND gate and shut off the second stage of the gate. It does this by generating another short negative pulse with a 74LS133 monostable multivibrator which clears the input flip flop CA1. The purpose of this feature is to stop the count when a maximum value is received and provide a means to alert the user that an out of limits or no signal is present. If the counters were allowed to continue counting, they would wrap around back to a zero value after the 65236th pulse was received and provide erroneous data to the user.

The remaining circuitry provides a data acknowledge output to alert the system controller that data is available on the system's data bus. A more detailed explanation of this asynchronous operation of the MC 68000 microprocessor will be described in the section on the system controller. The address of the bit timer is $2000.

Shift Generator

The shift generator's main function is to provide a positive going pulse to the shift selector at the calculated bit rate and in the center of the bit interval. For the purposes of shift generation, bi-phase-L signals are treated like NRZ-L. The shift pulses are modified in the shift selector to distinguish between NRZ and bi-phase type of PCM signals. Because the count obtained from the bit timer is digital in nature, the shift pulse cannot occur at exactly the bit rate. The result is a walking error due to the difference between the generated bit rate, and the exact bit rate. To minimize the walk error, the best possible time to read the bit is in the center of the

bit time. This is obvious since the walk error can be + or - 1/2 count. See the section on the walk error in part IV, Test and Evaluation.

The shift generator is one of the more complicated designs of the PCM processor (see figure 9). For the shift pulse to occur in the center of the bit interval, it must occur under certain circumstances. When a positive going signal pulse occurs, a shift must occur at half the signal time later to capture the bit. After that, and until the next positive signal pulse, a shift must occur at the bit rate in the center of the bit time as depicted in figure 4.

In order to meet these requirements, the shift generator is designed around two similar sections named the half counter and the one counter. The counters consist of two 74LS374 octal latches and four 74LS193 up/down counters each. The latches are used to load the count calculated by the system controller from the data read in by the bit timer. Data is loaded into the latches only once and remains there until the system is reset. After that, the logic loads the counters with the latch data as described below.

In figure 9, the top set of counters is the half counter and the bottom set is the one counter. Assuming that the half and one data is loaded into the 74LS374 latches, the operation is as follows. When a positive going signal enters the 74121 monostable multivibrator (AC3), a short 20ns negative pulse is generated and applied to the inputs of the two AND gates, CA2. This allows the latch outputs to load the counters with the count data. The 20ns pulse also sets the RS flip flop (AB3) causing the Q output to go high. This opens the AA5 AND

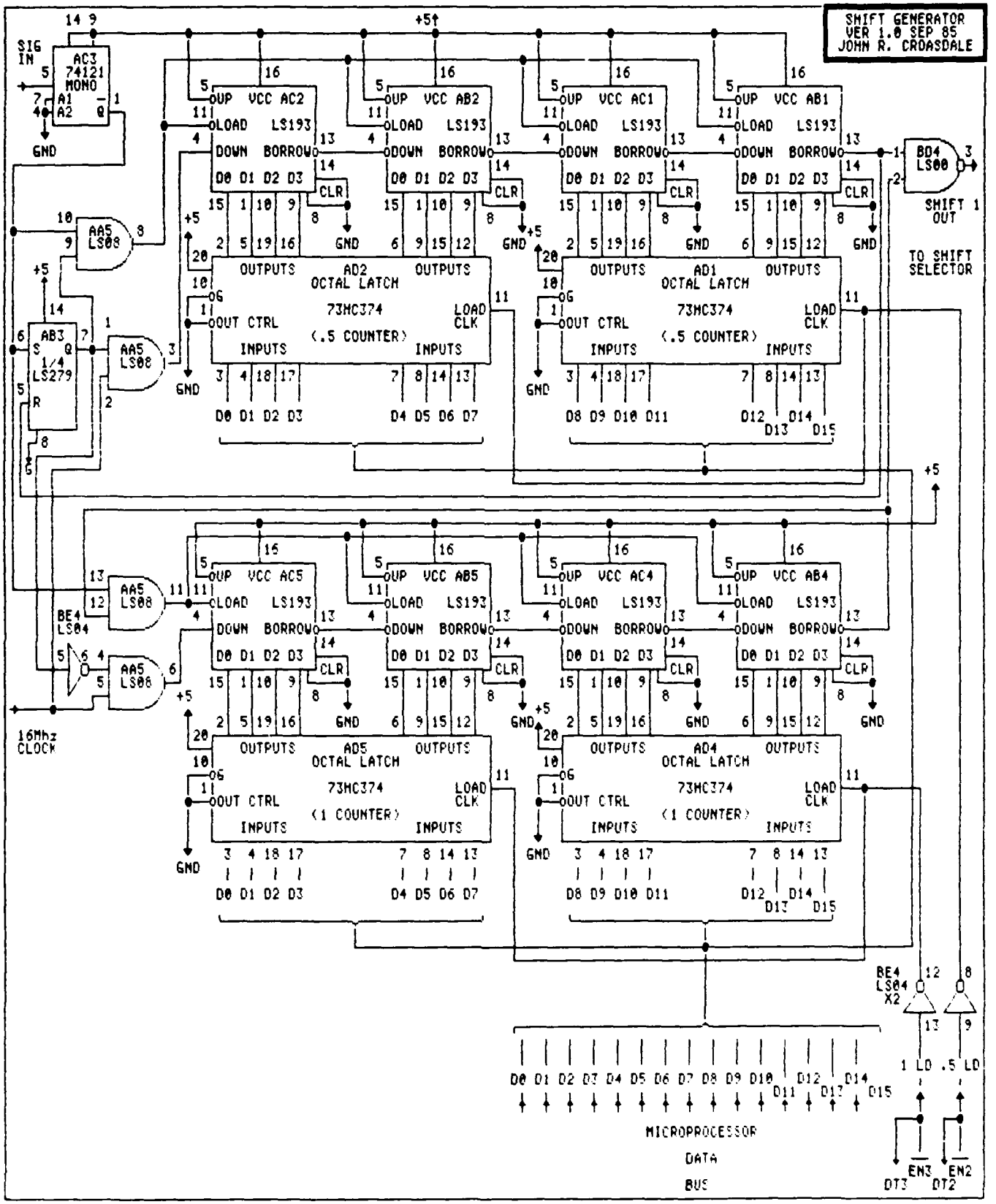


Figure 9. Shift Generator

gate and lets the 16Mhz clock through to the half counter. The Q output is also negated and applied to another AA5 AND gate which prohibits the clock from entering the one counter. Since the half counter contains the count for half the bit rate, a borrow will occur on AB1 half the bit time later and cause a positive going shift pulse to appear at the center of the first bit interval.

The borrow generated one half bit time later is then applied to the reset input pin of the RS flipflop. This action causes the Q output to go low and reverse the clock steering AND gates AA5. The half counter is now stopped but the one counter is receiving count pulses from the clock. The one counter contains the count for 1 bit time so the borrow output of AB4 will go low 1 bit time later. This borrow generates another shift pulse like the first one generated from the half counter. The borrow also causes the one counter to re-load and start over. This operation will continue to generate shift pulses at the bit rate until another positive going signal is input to the system.

The next positive going signal sets the RS flipflop again and the cycle is starts over. Using this technique, the system is re-calibrated at each positive transition of the input signal. If the signal consists of a long series of 1's or Ø's with no transitions in between, the walk error discussed in section IV may be encountered.

Shift Selector

The purpose of the shift selector is to modify the shift pulses so that they will be able to decode bi-phase PCM signals. The methodology

was designed around a simple observation concerning bi-phase signals.

Figure 10 shows a typical bi-phase signal. The shift generator generates shift 1 pulses as if the signal were NRZ. Shift pulses are shown as vertical bars at the bottom of the waveform. The value of the input signal during every other shift pulse (figure 11), will equal the value of the data bit. This is true only if the correct alternate bit is chosen. In figure 11, the correct bit has been selected. The shift selector's task is therefore to forward the correct alternate shift pulses to the word processor.

The task of halfing the shift 1 pulses to make shift 2 pulses is accomplished by a 74LS74 D type flip flop (see figure 12). The 74LS74 is configured as a toggle flip flop which changes state at the occurrence of a shift pulse at pin 3. The toggle action is obtained by connecting the Q' output to the D input. The output of the flip flop is not in the correct form of the pulse needed by the word processor. The output instead, is a square wave which must be converted to a short negative going pulse. The NAND gates (BD4), perform both this function. These gates also select which type of PCM (bi-phase or NRZ) is to be decoded.

For NRZ, the first BD4 NAND gate is shut off not allowing the output from BA5 to get through. For bi-phase, the gate is open and the output of BA5 reaches the second BD4 NAND gate. The input signal is also fed to this NAND gate where it is used to shape the square wave from BA5 to a short negative going pulse which is applied to the word processor.

The job of synchronizing the incoming series of shift pulses so

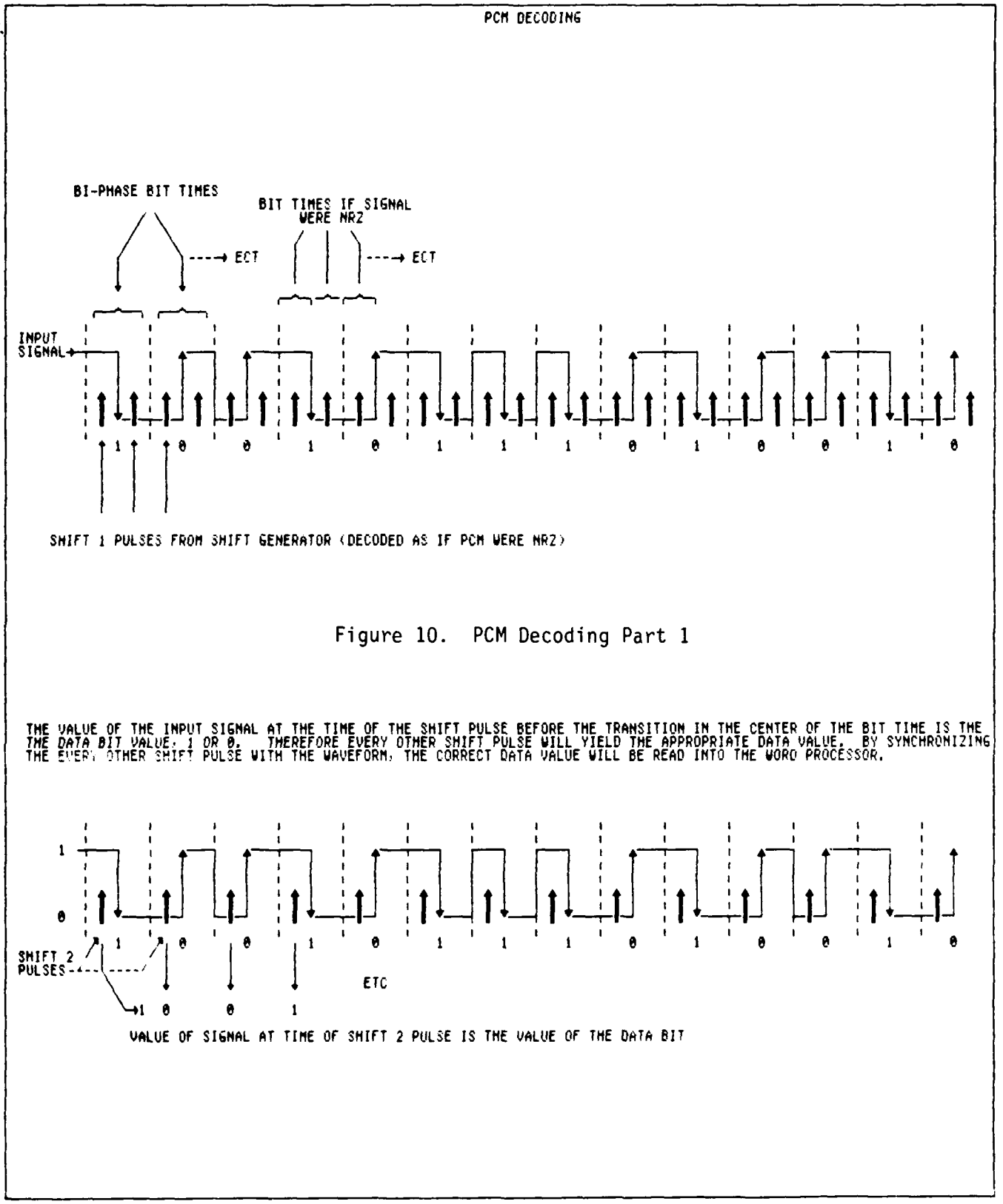


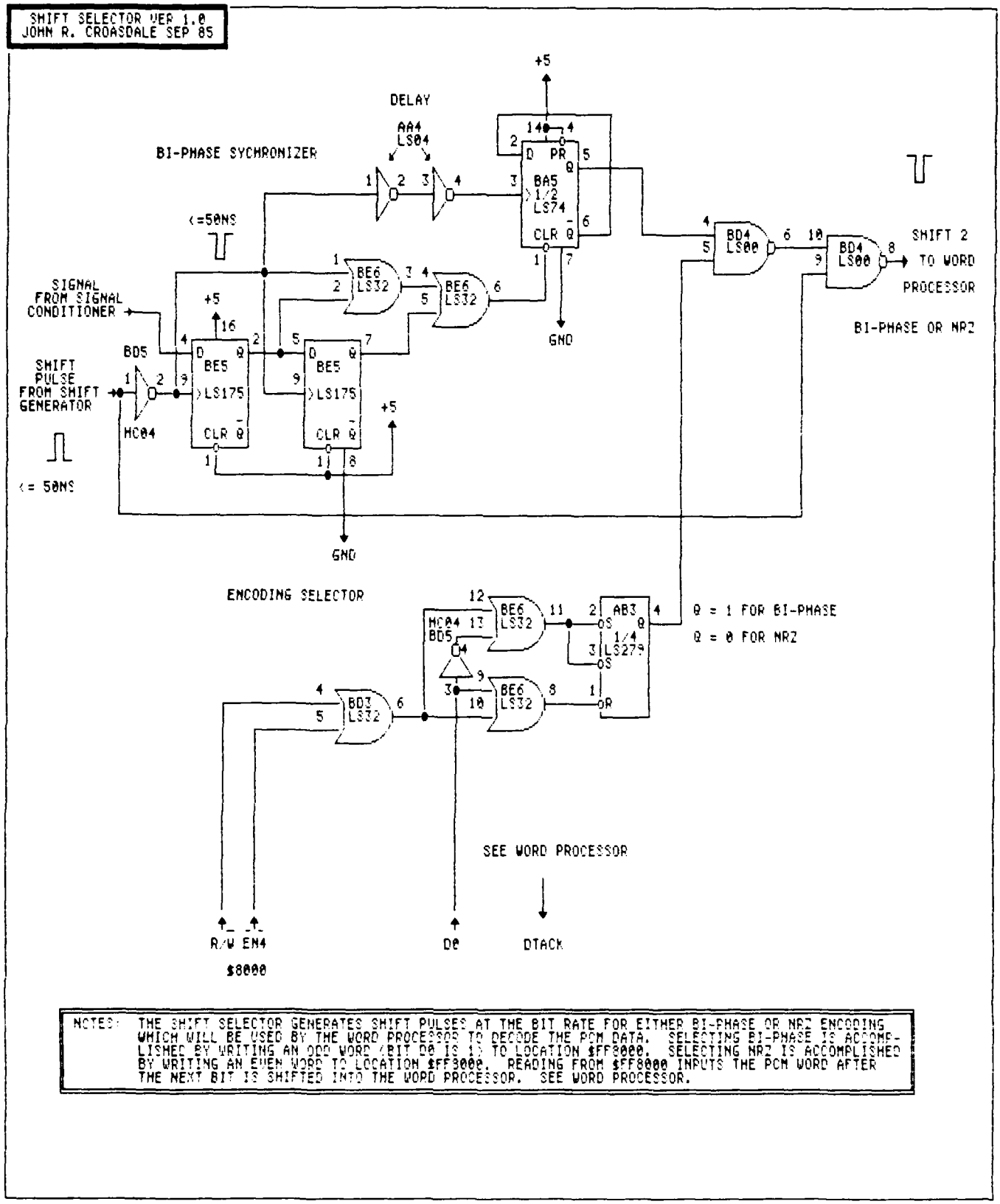
Figure 10. PCM Decoding Part 1

THE VALUE OF THE INPUT SIGNAL AT THE TIME OF THE SHIFT PULSE BEFORE THE TRANSITION IN THE CENTER OF THE BIT TIME IS THE DATA BIT VALUE, 1 OR 0. THEREFORE EVERY OTHER SHIFT PULSE WILL YIELD THE APPROPRIATE DATA VALUE. BY SYNCHRONIZING THE EVERY OTHER SHIFT PULSE WITH THE WAVEFORM, THE CORRECT DATA VALUE WILL BE READ INTO THE WORD PROCESSOR.

Figure 11. PCM Decoding Part 2

that the correct phasing of BA5 is accomplished is done by the two 74LS175 D type flip flops, BE5. As mentioned above, the incoming shift pulses are generated as if the signal were NRZ. When this is accomplished, a bi-phase "1" would decode to a "1,0" in NRZ. This is true since a transition from high to low is considered a "1" in bi-phase-L. Likewise, a bi-phase "0" would be a "0,1" in NRZ decoding. Disallowed values are "0,0" and "1,1". These values occur all the time but they straddle the bit interval boundaries instead of occurring during them (see figure 13). If two NRZ decoded values of "0,0" occur, the next NRZ decoded value must occur in the second half of the bi-phase bit interval, and must be the inverse of the bi-phase bit value. The same reasoning is valid for an NRZ decoded bit pattern of "1,1". By looking at the input signal at the time of the next shift pulse after the "0,0" or "1,1" bit patterns, the data bit value can be obtained. By looking at the signal during the second "0" or "1" shift pulse, and every other shift pulse after that, the data can be obtained directly. The circuit shown in figure 12 does just that.

To accomplish this function, the trailing edge of a negated incoming shift pulse clock the signal bit into the first D flip flop. The 74LS175 clocks on a positive going transition so the shift pulse must be negated. The shift pulse is significantly shorter than to the input signal, approximately 1/200th for a 100Khz signal, so the signal level will still be valid after the shift pulse is over. The bit must be shifted in on the trailing edge to avoid race problems and to allow all transitions to stabilize before the next shift pulse occurs. The



NOTES: THE SHIFT SELECTOR GENERATES SHIFT PULSES AT THE BIT RATE FOR EITHER BI-PHASE OR NRZ ENCODING WHICH WILL BE USED BY THE WORD PROCESSOR TO DECODE THE PCM DATA. SELECTING BI-PHASE IS ACCOMPLISHED BY WRITING AN ODD WORD (BIT D0 IS 1) TO LOCATION $FF8000. SELECTING NRZ IS ACCOMPLISHED BY WRITING AN EVEN WORD TO LOCATION $FF8000. READING FROM $FF8000 INPUTS THE PCM WORD AFTER THE NEXT BIT IS SHIFTED INTO THE WORD PROCESSOR. SEE WORD PROCESSOR.

Figure 12. Shift Selector

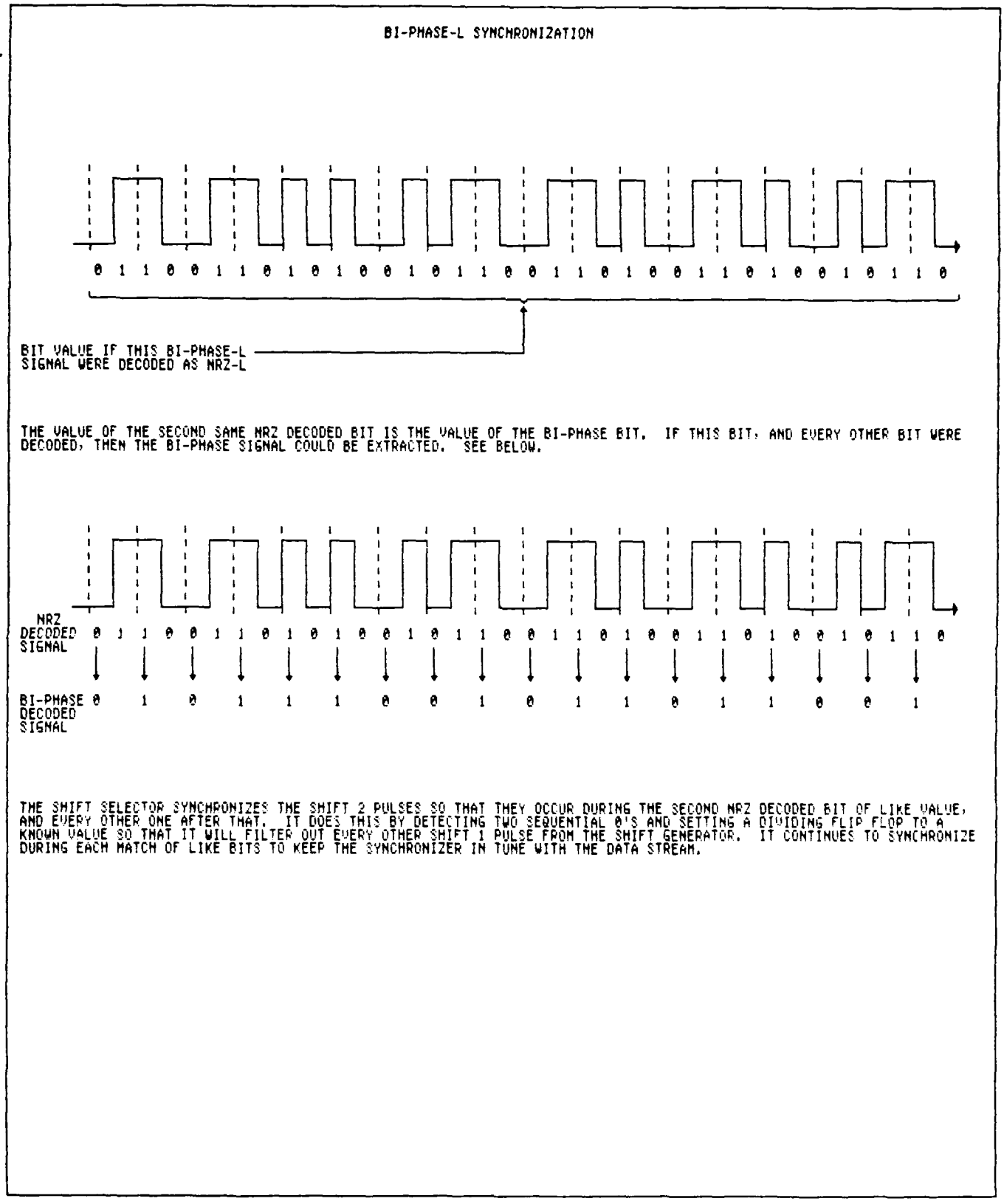


Figure 13. Bi-Phase-L Synchronization

next shift pulse clocks the bit into the second flip flop, and the next bit into the first. In this way, two successive bits are continuously monitored. When a pattern of "Ø,Ø" occurs, both OR gates, BE6, are open, awaiting the next shift pulse.

When the next negated shift pulse occurs, the BA5 flip flop is cleared to a known state. At the trailing edge of the negated shift pulse, and after a slight two gate delay to avoid race problems, BA5 is toggled back to where the Q output is a "1". It can be noted that at any time Q is high, the next shift pulse will not get through the BD4 NAND gate and will be deleted from the shift pulse stream. It will therefore take the next two shift pulses to open the gate and generate the output pulse denoted as shift 2. By observation, this is exactly the pulse which occurs at the time of the valid bi-phase data bit. All the word processor needs to do is to clock in the incoming signal using the generated shift 2 pulse.

The PCM type, NRZ-L or bi-phase-L, is selected by writing either a "Ø" or "1" into the SR latch, AB3. This is accomplished by writing an even data word to address location $FF8000 for NRZ decoding, or an odd data word to the same location for bi-phase. When EN4 is high, or R/W' is high, a high is applied to all inputs of of the latch. This designates a no change condition at the output. The only time that both EN4 and R/W' are low is when the system controller writes to location $FF8000. When this happens, BD3's output goes low, opening the BE6 OR gates. This allows the least significant data bit, DØ, to reach the inputs of the latch. If DØ is low, the reset input goes low causing the Q output to go low to select NRZ decoding. If DØ is high,

the set input is toggled low causing the Q output to go high and select bi-phase. The system controller guarantees that both R/W' and DØ are valid before EN4 and are held valid until after EN4. This assures that no race problems or glitches occur during the writing operation.

It should be noted that reading from $FF8000 affects the word processor discussed below.

Word Processor

The word processor consists of two sections. The first section performs the serial to parallel conversion required of the PCM monitor's PCM processor. The second section does the bits per word counting.

The serial to parallel conversion is simple because most of the work has already been accomplished by the SIGNAL CONDITIONER, shift generator, and shift selector. The circuit, see figure 14, consists of two 74LS299 tri state output shift registers. The clock is the shift 2 pulse from the shift selector, and the input is the input signal. Each time a shift 2 pulse arrives, a new bit from the signal is shifted in from the left or right depending on the position of the PCM order switch, SW2. The PCM order can be most significant bit first (MSB) or least significant bit first (LSB) and must be set by the user before using. When this occurs, the bits already in the register are shifted one bit position, left or right depending on SW2. SW2 controls the select inputs to the shift registers, S1 and S2, and thus tells the register which way to shift. The signal, however, must be applied to both ends of the register pair as shown in figure 14.

As mentioned in the overview, there are several ways to read the data. Data can be read bit by bit so that a frame synchronization or some other pattern can be found. Data can also be read in one word at a time for reading the actual data words. The key is to assert the data acknowledge (DTACK) signal at the correct time to specify that the data is valid and the system controller can continue processing. See section on the system controller.

When reading data at the bit rate, the DTACK signal (DT4) must be asserted at the bit rate and at the right time. Since data is valid during the shift 2 pulse, it is used to trigger the lower left AB3 SR latch. The Q output of this latch is held high until EN4 is asserted low. This makes the set inputs high, releasing the latch. The next shift 2 pulse triggers the reset input low which makes the Q output, and likewise DT4, go low. The latch will remain in this state until EN4 again goes high, toggling the latch output high again. Also, when EN4 goes low, the EN2 enables of the shift registers are asserted through the 74LS02 NOR gate. To complete the enables, the R/W' line must be high for EN1 to be asserted.

The action is summarized as follows. First the system controller reads from location $FF8000 causing the R/W' line to go high, and EN4 to go low. This action enables the outputs of the shift registers and applies them to the systems data bus. When the next shift 2 pulse occurs, AB3 is toggled causing DT4 to be asserted and tell the system controller that data is valid. The system controller then captures the data from the data bus and removes EN4. This causes DT4 to return high where it will remain until another bit read cycle is encountered.

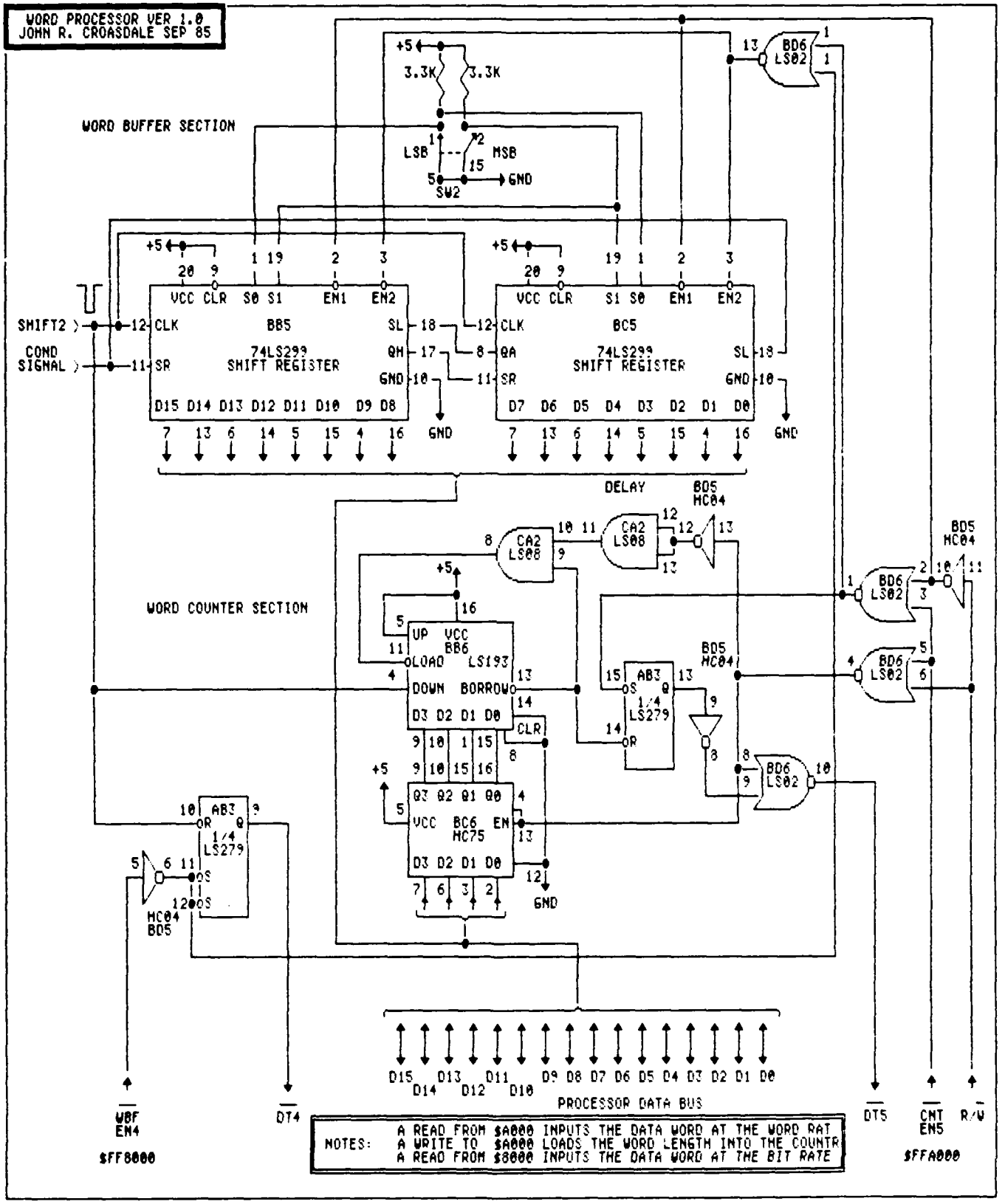


Figure 14. Word Processor

Reading data at the word rate is accomplished by reading from location $FFA000. Since the word length must be able to vary according to customer needs, another stage in the word processor is needed. This stage counts bits. When enough bits have been counted according to the word length, it tells the system controller that a word is now in the shift register.

Reading from $FFA000 causes the R/W' line to go or remain high, and the EN5 enable line to go low similar to the bit read function above. To perform the read function, the shift registers must be enabled, and a DTACK, in this case DT5, must be asserted. The R/W' line is inverted which causes EN1 to go low just as before. In addition, it causes output pin 4 of the NOR gate, BD6, to go low. This always occurs regardless of the level of EN5. At the same time, the pin 1 output goes high enabling the shift registers through EN2, and releasing the SR latch AB3. When the 74LS193 down counter BB6 times out, a borrow is asserted low causing the SR latch Q output to go low and also DT5 through an inverter, and NOR gate. The borrow signal is extremely short so the reset line returns high almost immediately. The system controller, after receiving DT5, will release EN5 high and cause the SR latch set input to go low. This causes Q to go high along with DT5 in turn.

Counter BB6 must have the correct bit count loaded in for the system to work. This is done by writing the word length into the quad latch BC6 where it will be stored indefinitely. The circuit also loads the counter to initialize it after a frame sync. See software section. Loading the latch and counter are done by writing to location $FFA000.

When this happens, the R/W' line will go low along with EN5. Two lows at the input pins 5 and 6 of the BD6 NOR gate will cause its output to go high. This will cause DT5 to go low and alert the system that data has been read. While this occurs, the quad latch, BC6, is enabled. The enabling signal is also fed, after being inverted by BD5 and delayed by CA2 to allow the Q outputs of BC6 to stabilize, to an AND gate, the second CA2. This low input to the AND gate will cause its output to go low and load the counter, BB6.

Once loaded, the counter will count shift 2 pulses until a borrow is generated. The borrow then re-loads the counter for the next word cycle, and, as mentioned above, resets the SR latch to assert DT5.

Writing to $FF8000 affects the shift selector as explained in the previous section. The maximum count that can be loaded into the word processor is 16, $000F hexadecimal. The least significant 4 bits of the data word, D3 - D0, should contain the bits/word count.

System Controller

The system controller's function is to control the PCM processor functions of the PCM monitor. It executes the software that performs automatic functions such as the calculation of bit rate and PCM type. The system controller is based on an MC68000 16 bit microprocessor because of its asynchronous operation, speed, ease of interfacing to, and flexibility. Figure 15 depicts the schematic of the system controller. The processor is used in a very basic manner and takes advantage of its zero page short addressing and asynchronous I/O capabilities. Because the controller is used in a special application

and not as a general purpose computer, address decoding is kept to a minimum to avoid an excessive chip count. In fact, the memory map shown on the next page consists of only 8 sections called pages.

The Memory Map, as shown is arranged into 8 pages. Therefore only 3 address bits are decoded, A15, A14, and A13. The 74LS138 decodes these three lines and enables one of 8 areas of memory. These pages are described below.

Page Ø, the program ROM area, is assigned address from $000000 to $001FFF. This area spans 8K bytes or 4K words. Each instruction's operation code (OP-CODE) must reside at an even address for the processor to function correctly. Because only 16 bit words will be used, byte addressing is not allowed. This means that only even addresses can be accessed. See section below for more explanation on addressing. The program memory chip currently used is a 2K by 8 2716 EPROM. The design allows for a 2732 4K by 8 or a 2764 8K by 8 EPROM to be used if expansion is desired. The 2764 must be manually switched to select which bank of 4K is desired since only 4K words of address space is dedicated to ROM.

The EPROM is very simple to access. Address lines A1 thru A13 are connected in parallel to the ROM HI and ROM LO chips. AØ is an internal address line to the MC68000 processor and is used to address bytes. In this implementation, byte addressing is not used so AØ will always be at a low logic level or tri-state as the situation dictates. Data lines DØ thru D7 are connected to the data lines of ROM LO and D8 thru D15 are connected to ROM HI. All enable pins are connected together to form a common enable line for the ROM memory. Since ROM

PCM monitor MEMORY MAP

ADDRESS	USED FOR	REMARKS
$FFFFFE		
	RANDOM ACCESS MEMORY	2K words, use $FFF800
$FFE000		to $FFFFFE
$FFDFFE		
	I/O DUAL PORT MEMORY	1K words, use $FFC000
$FFC000		to $FFC7FE
$FFBFFE		
	WORD LENGTH COUNTER	1 word, use $FFA000
$FFA000		
$FF9FFE		
	WORD BUFFER	1 word, use $FF8000
$FF8000		
$007FFE		
	ONE COUNTER	1 word, use $006000
$006000		
$005FFE		
	HALF COUNTER	1 word, use $004000
$004000		
$003FFE		
	BIT TIMER	1 word, use $002000
$002000		
$001FFE		
	PROGRAM MEMORY	2K words, use $000000
$000000		to $000FFE

NOTE: 1. Address in the MC68000 are BYTE address. Therefore 2K words will span 4K of address space etc.

2. Zero page includes the first 32K bytes and the upper 32K bytes in an address range of 16.8 Megabytes. Therefore address from $000000 to $007FFF and $FF8000 to $FFFFFF are considered zero page.

Table 1. PCM PROCESSOR MEMORY MAP

can only be read from, the enable line must not be valid during a write cycle. This is accomplished by negating the read/write line (R/W') from the processor, and OR'ing it with the ROM enable line from the 74LS138 address decoder. If the processor tries to write to the ROM memory, the enable line will be high causing the ROM to be deselected.

In addition to enabling the ROM memory, the ROM enable line from the decoder starts a small counter to delay the data acknowledge line from signaling the processor that the data is ready. This is needed because the access time of the 2716 (350ns to 450ns) is much slower than the processor (2.5 clock cycles, 312ns for 8Mhz or 156ns for 16 Mhz). This circuit counts 4 clock cycles and then drives DTACK low. When this happens, 2 clock cycles later, the processor latches the data into an internal buffer and the read cycle is terminated. The count of 4 was chosen to allow for the expected worst case.

Page 1 resides from memory locations $002000 to $003FFE and is used to access the bit timer. Since the bit timer is only one location, a read from any one of the 4K locations will result in a successful read. For consistency sake, address $002000 has been chosen to access the bit timer.

Page 2 resides from memory locations $004000 to $005FFE and is used to program the half counter section of the shift generator, explained later in this report. The processor can either read or write to this location without problems but a read will be meaningless. Again, the half counter section is only one location so the processor can write to any one of the 4K locations with the same effect. Address $004000 has been chosen for the half counter to be consistent.

Page 3 resides in the next 4K of memory space, $006000 to $007FFE and is used to load the one counter of the shift generator. Like the half counter, any of the 4K address are valid but $006000 will be used.

Page 4 uses locations from $FF8000 to $FF9FFE to access the word buffer and the shift selector. Writing an odd word to to page 4 causes the shift selector to select Bi-Phase decoding. Writing an even word causes it to select NRZ. Reading from page 4 causes the word processor to output the data word at the bit rate. Address $FF8000 will be used to write and read as described above.

Page 5 comprises locations $FFA000 to $FFBFFE. Writing to this location loads the word length into the word processor. Reading from this location loads the data word at the word rate (versus the bit rate from page 4) into the system controller. Address $FFA000 will be used for these functions

Page 6 is the I/O RAM port consisting of two dual port 1K by 8 static RAMS. Page 6 consists of locations $FFC000 to $FFDFEE but since the I/O RAM is only 2K bytes long (1K words), only address $FFC000 to $FFC7FE will be used. Although only intended to be written to, the I/O PORT has certain reserved locations to pass control parameters from and to the display processor. See software section for this description. The RAM chips are dual ported and transparent to both the controller and to the display processor without any arbitration logic needed.

Page 7 is the RAM area. Two 2K by 8 TMM2016 static RAMs are used to provide 2K words of fast memory. The program is transferred to this RAM from the ROM where it can run at maximum speed. The stack for the MC68000 processor and a scratch pad area also reside here. The top 8K

locations of the address space, $FFE000 to $FFFFFE are accessible but only $FFF800 to $FFFFFE will be used in this implementation.

Some other features of the system controller should be explained. The MC68000 microprocessor has an elaborate interrupt and error exception processing capability. None of these will be used because they are not needed. As a result, the exception inputs are tied high. These inputs are IPL, BR, BGACK, VPA, BERR and HALT.

The RESET and HALT lines are connected to an RC network to reset the processor upon power up. The values were chosen to allow enough reset time (at least 100 milliseconds upon power up) to do the job.

As mentioned earlier, each I/O device must return a data acknowledgement when data is valid or has been read. Each of these lines, DT1 thru DT7 plus the ROM DT line from the cycle counter, is NOR'ed and applied to the DTACK line of the processor. The combination of the NAND gate (BE3) and the inverter (BE4) acts like a NOR gate for negative logic and applies a negative signal to DTACK when one of the DT lines goes low. Otherwise, DTACK is high.

The address decoding logic would normally operate using the ADDRESS STROBE (AS), UPPER DATA STROBE (UDS) and LOWER DATA STROBE (LDS) lines as required. The AS line from the processor is valid (low) when a valid address is present on the address bus. This normally tells the decoding logic that the address is valid. To keep the chip count as low as possible, it is not really necessary to use the AS line because the UDS and LDS lines operate in much the same way when only word addressing is allowed.

For a read operation, the UDS, LDS, or both are valid (low) at the

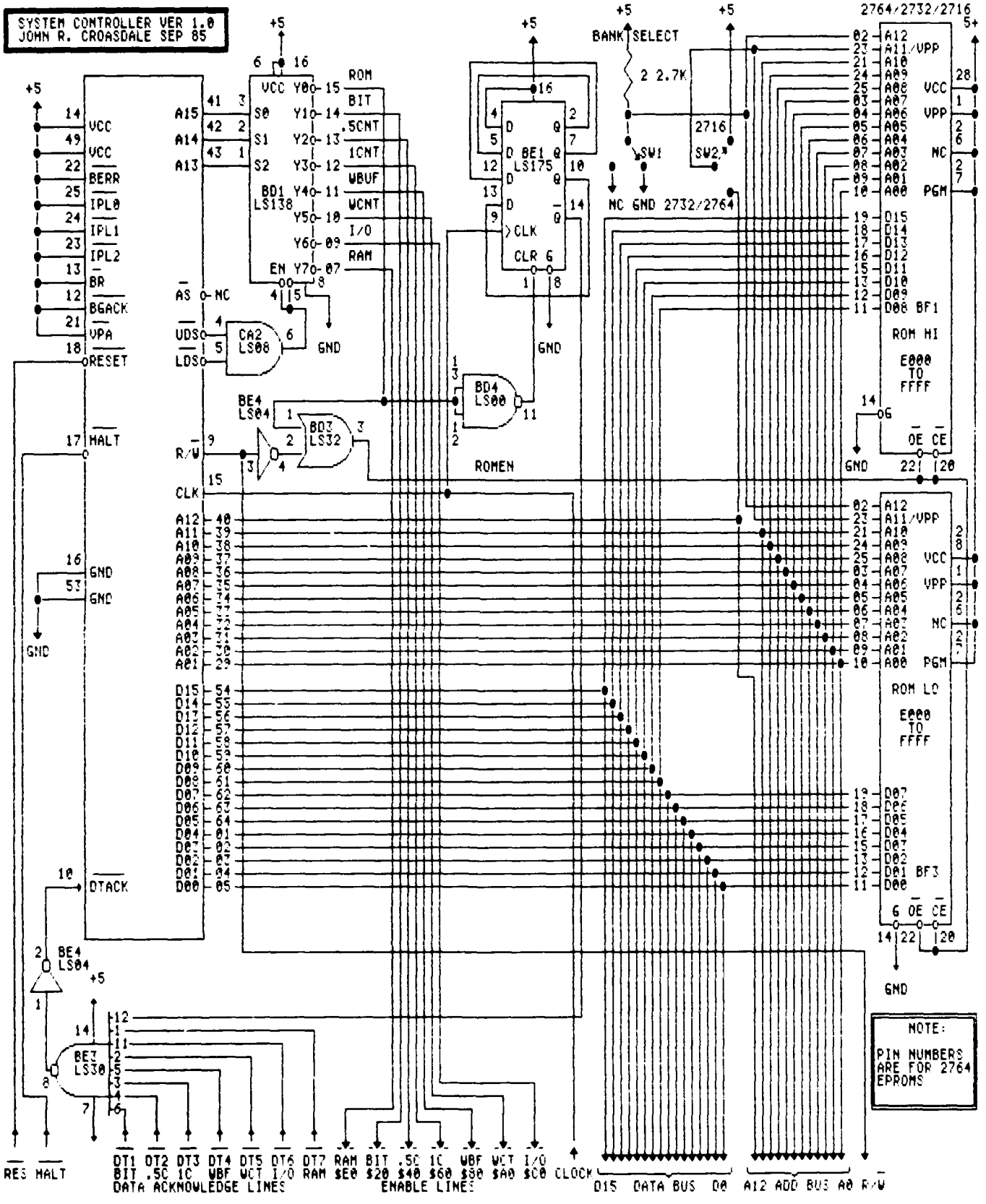


Figure 15. System Controller

same time as the AS line. UDS and LDS actually reflect the state of the internal address line, A0 and are used for byte operations. Because only word addressing is allowed, both UDS and LDS lines are valid at the same time. In this case, the AS line, UDS line, or LDS line can be used to tell the decoder (BD1) that the address is valid. The OR'ed combination of UDS and LDS was chosen because of write cycle considerations explained below.

The MC68000 timing diagrams show that during a write cycle, the AS line is valid one clock cycle before the data is valid (valid UDS and LDS). This was designed into the processor to give the address decoder sufficient time to decode the address during a write operation. Because of the simple and fast decoder design utilized in the system controller, this excess time is not needed. A consideration exists during a write cycle if the data acknowledge line, DTACK, is asserted too early in the cycle. This is not critical for the ROM section because the DTACK signal is delayed 4 clock cycles anyway. It does make a difference, however, for the shift generator and word counter where DTACK is generated directly from the decoder's output lines. For these sections, DTACK occurs only a few nanoseconds after the decoder is enabled. If the AS signal were used as an enable, it would cause non-valid data to be written. The UDS or LDS lines, however, are asserted when data is valid and provide a better enable signal. Both UDS and LDS are OR'ed together so that either one can enable the decoder. This prevents possible hang up if an error in programming results in a byte access.

As mentioned earlier, the system controller is a very simple

utilization of the MC68000 microprocessor. Used in this configuration, high speed, asynchronous I/O access, and minimal chip count is realized. It also allows for easy system debugging and modification. If at a later date expansion is required, further address and exception processing logic can be added at the expense of speed and complexity.

Reset Circuitry

The reset circuitry is fairly simple. The circuit causes the initial power-on reset and allows the user to reset the system manually. To reset the MC68000 on power-up, the reset line and the halt line on the processor must be brought low for at least 100ms. For a system reset, only the reset line must be brought low. Consult figure 16.

A 555 timer integrated circuit in the monostable configuration performs the power-on reset function. When the power is first turned on, the trigger at pin 2 is held low causing the output at pin 3 to go high. Also, C2 begins to charge through R2. Since the trigger is connected to a smaller RC network (R1 and C1), it will reach a high state before the threshold voltage and allow the system to time out. Time out occurs when the threshold voltage at pin 6 of the 555 reach 2/3 of 5 volts. When this happens, the output goes low and remains there until the system in turned off then on again. The time to charge C2 to the threshold is approximately 1.1*R2*C2, or in this case, 517ms.

The output is negated by one section of CC4 and applied to the halt line of the processor. It is also NOR'ed with the manual reset circuitry. When the output is high, the reset line is low causing the

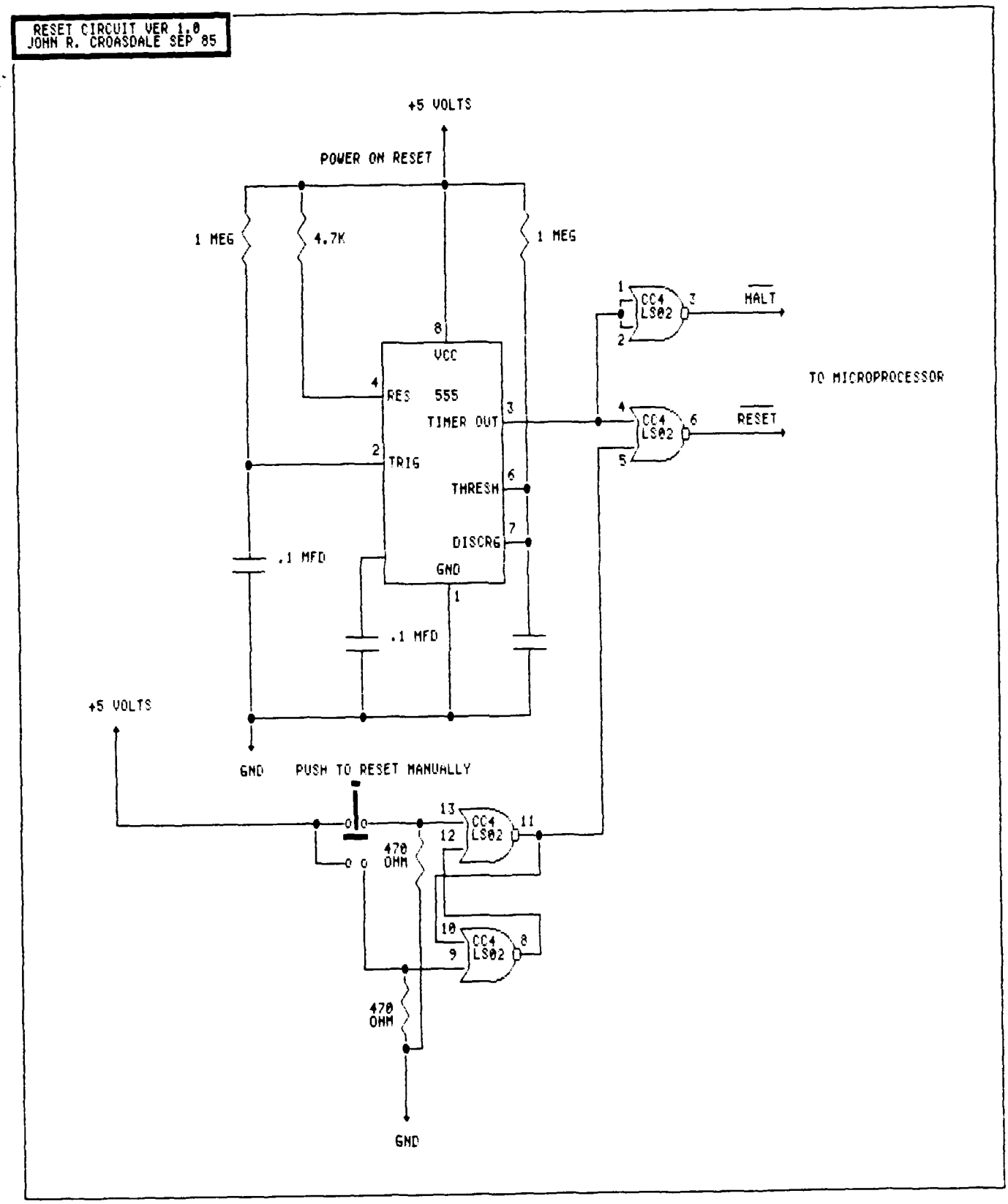


Figure 16. Reset Circuit

processor to perform the reset function.

Manual reset is accomplished by two sections of CC4 configured as an RS latch. When the reset button is pressed, pin 9 of CC4 goes high causing pin 8 to go low. Also pin 13 is pulled low by a resistor. Two lows at the input of a NOR gate cause its output to go high. This output is applied to the NOR gate as mentioned in the previous paragraph. When the button is released, the latch resets back letting the output go low as before. The user must keep the button depressed for at least 1/10th of a second for a system reset which is no problem except for the very quick.

Random Access Memory

The random access memory section of the PCM monitor is very simple and consists of two, 2K by 8 static RAMs (see figure 17). Each RAM is connected to the address bus, A0 thru A10, and the respective portions of the 16 bit data bus. The output enable is grounded so that it will be valid when the RAM is selected by accessing addresses $FFE000 to $FFE7FE. As mentioned earlier, the MC68000 requires a DTACK signal when data is valid. This acknowledgement is shown on the system controller schematic and is explained above. The TMM 2016 static RAM chip has an access time of 250ns so the delay required by the ROM is not needed for the RAM.

The RAM is used to provide an area for fast program execution to enable the PCM processor to perform at its maximum capability. It is also used as a program stack for the MC68000 processor and as a scratch pad for calculations needed by the system software.

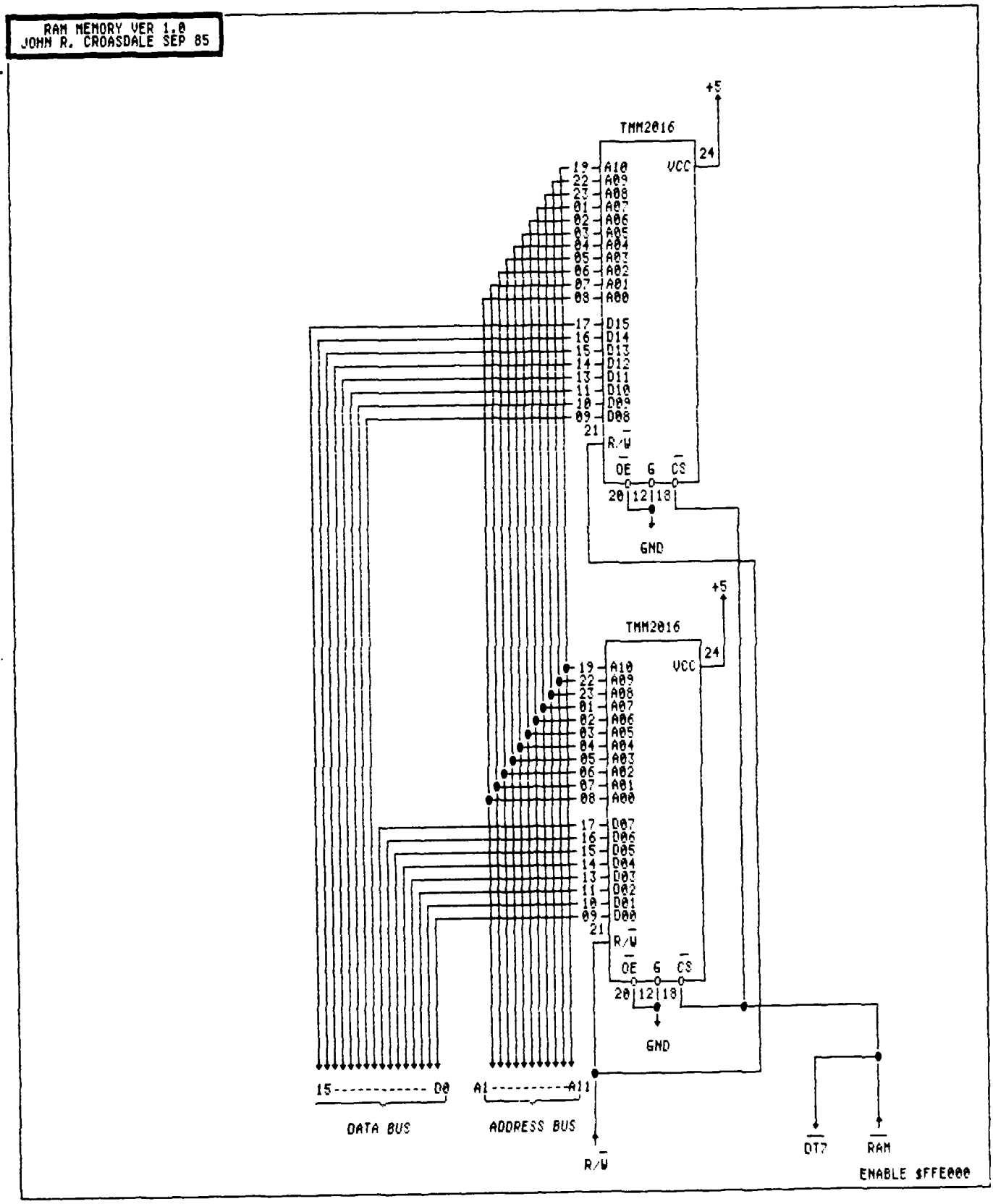


Figure 17. RAM Memory

I/O Port Ram

Figure 18 shows the I/O PORT schematic. The advent of the IDT 7130 dual port 1K by 8 bits RAM chip makes the interface of two running computers or processor very easy. The chip has two sets of address, data, and control lines and allows each set to be accessed asynchronously by two different systems.

Each system treats the I/O PORT as a static RAM which, in fact, it is. The PCM processor treats the 7130 as 1K 16 bit words mapped into locations $FFC000 to $FFC7FE. The display processor treats the 7130 as 2K 8 bit words mapped into Color Computer locations $E000 to $E7FF. See the display processor section below. The 7130 operates the same way as the random access memory explained above.

Power Requirements and Design

The PCM monitor requires a single 5 volt regulated power supply capable or producing 2 amp with no more than 100mv of ripple. The system itself draws only about 1.2 amps in its current configuration but there must be room for expansion.

The unit is constructed on a 6 by 10 inch wire wrap board so it can be easily modified. In its final configuration (separate effort) it will be mounted on a printed circuit card, the size of which is unknown at this time. It will be compatable with the bus structure adopted by the 4950th Test Wing so that it could easily be added to current computer systems if required. See appendix E for a parts list.

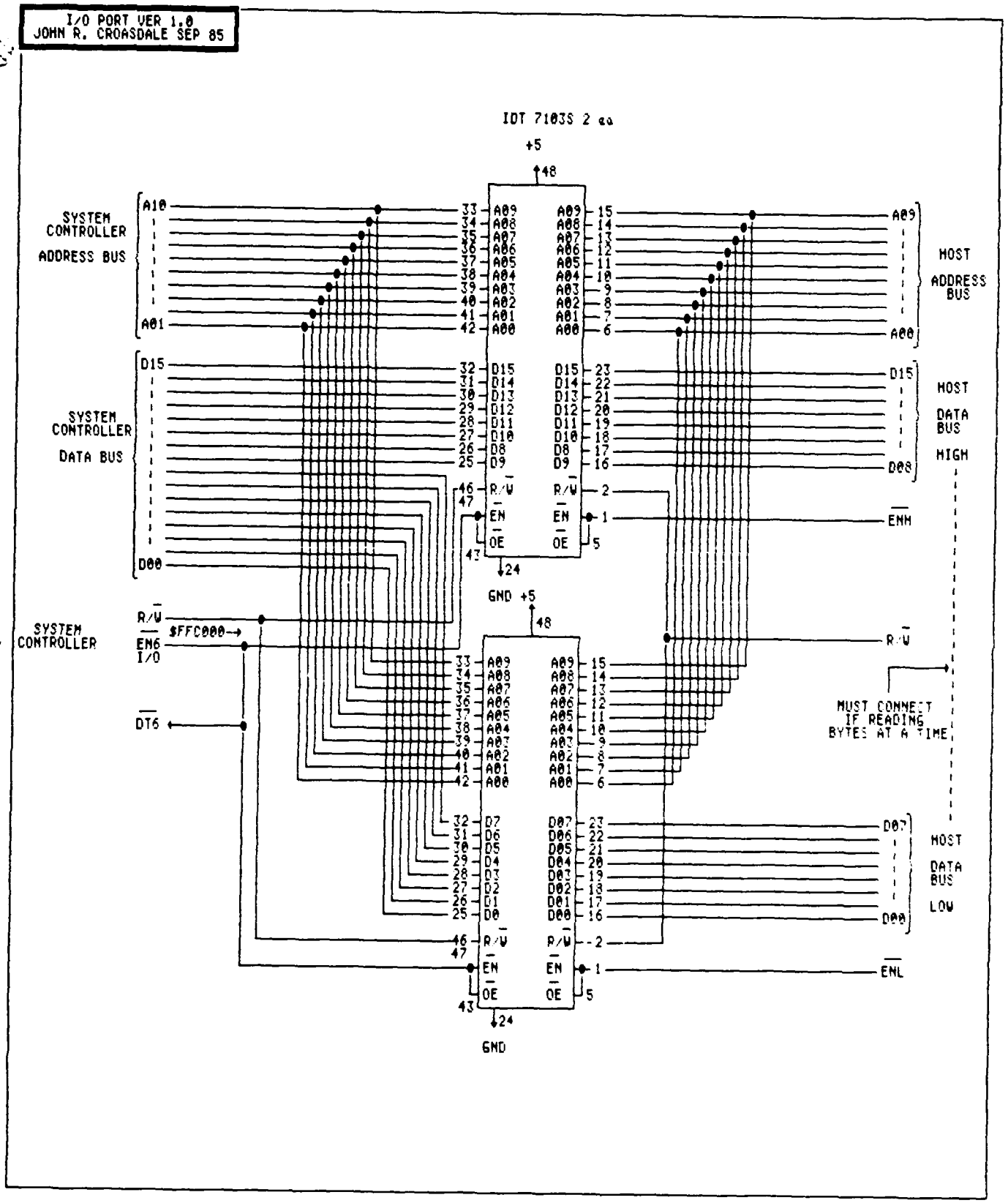


Figure 18. I/O Port

Software

The programming of the PCM processor is done in machine language to allow the real-time interactions with the PCM data stream at bit rates up to 100KBPS. The system must first interact with the display processor to allow a user to control its different special features. These features are described in the display processor software sections and listed below.

1. Auto bit rate calculation.
2. Manual bit rate entry.
3. Auto PCM type calculation.
4. Manual PCM type calculation.
5. Normal polarity.
6. Inverse polarity.
7. Variable word length of 8-16 bits.
8. Continuous mode of operation.
9. Trace mode of operation.

The user controls these features through the I/O control block located at addresses from $C7E0 through $C7FE with respect to the PCM processor. These addresses map into the display processor at locations $E7E0 through $E7FE because of the independent nature of the dual port RAM. Each system sees the RAM independently and can therefore map it anywhere it needs it. The I/O control block is just a few memory locations used to pass information back and forth between the two systems. See table 2 for a listing of the I/O control block locations.

The most important location in the I/O control block is the valid flag. The PCM processor has two modes of operation, continuous and trace. With this flag, the user tells the PCM processor to start one of these operations, or reset.

The continuous mode of operation provides the user with a

SOFTWARE MAP
FOR PCM monitor

$F800 - $FFFF RAM
$C000 - $C7FE I/O RAM

$C7E0 - $C7FE I/O CONTROL BLOCK

I/O CONTROL BLOCK DEFINITIONS

$C7FE	valid FLAG	INPUT: 0=continuous MODE 1-00FF=TRACE MODE, NUMBER=DATA WORD > $7FFF STOP AND WAIT
$C7FC	USER BIT RATE	INPUT: 0=AUTO, <>0=RATE IN WORD COUNT
$C7FA	PCM TYPE	INPUT: "C"=CALC, "N"=NRZ, "B"=BI-PHASE
$C7F8	WORD LENGTH	INPUT: HEX VALUE 8 - 16 BITS
$C7F6	FRAME LENGTH	INPUT: HEX VALUE 0 - $FFFF
$C7F4	FRAME SYNC1	INPUT: HEX VALUE
$C7F2	FRAME SYNC2	INPUT: HEX VALUE
$C7F0	FRAME SYNC3	INPUT: HEX VALUE
$C7EE	ERROR COUNT	OUTPUT: NUMBER OF SYNC ERRORS
$C7EC	POLARITY	INPUT: "N"=NORMAL, "I"=INVERSE
$C7EA	FRAMES/BUFFER	INPUT: HEX VALUE
$C7E8	CALCULATED TYPE	OUTPUT: "N"=NRZ, "B"=BI-PHASE
$C7E6	CALCULATED RATE	OUTPUT: RATE IN WORD COUNT
$C7E4	CORRECTION	OUTPUT: NUMBER ADDED TO COUNT TO SYNC IN
$C7E2	STATUS	OUTPUT: 0=NOT IN SYNC, <>0=IN SYNC
$C7E0	DATA ADDRESS	OUTPUT: ADDRESS OF LAST DATA VALUE WRITTEN

$C000 - C7DE OUTPUT: DATA

$A000 WORD LENGTH COUNTER
  WRITE = STORES WORD LENGTH
  READ = DATA WORD 16 bits at word rate
$8000 WORD BUFFER
  WRITE ODD = BI-PHASE
  WRITE EVEN = NRZ
  READ = DATA WORD 16 bits at bit rate
$6000 ONE COUNTER
$4000 HALF COUNTER
$2000 BIT TIMER
$0000 - $0FFE PROGRAM EPROM

TABLE 2 PCM PROCESSOR EXPANDED MEMORY MAP and I/O CONTROL BLOCK

continuous update of inputted PCM data regardless of whether or not the display processor can read it all. If the display processor is slow, data will be lost as the PCM processor will keep writing data to the buffer whether it is read out or not. In other words, it is up the the display processor to use the data as it sees fit.

trace mode, on the other hand, is synchronized to the display processor in that it fills the memory buffer with sequential frames of only one data word. When the buffer is full. the PCM processor remains in sync but stops storing data. A user input telling it to begin another trace by resetting the valid flag with another data word value, or the same one will start the trace over again. If the valid is negative, the system resets and is returned to the continuous mode. While in trace mode, the display processor can read each and every frame of one data word and use the information to catch glitches etc. This process is described also in the display processor software section and in Appendix B.

Upon startup, the PCM processor sets the valid flag to all F's, all bits high. When the display processor inputs all the PCM stream data into the I/O control block, it signals the PCM processor to start looking for the PCM signal by setting the valid flag to Ø. If it wants it to stop, the valid flag must be set to some negative number (bit 15 must be set). If the trace mode is desired, the display processor must set the valid flag to some positive value. That value must be the number of the data word in the PCM stream to do the trace on. No error checking is accomplished and if a data word number larger than number of words per frame is entered, the system will lose sync and reset.

The contents of the user bit rate location determines whether the user wishes to use an automatic calculation of the bit rate or not. Placing a Ø in this location will force the PCM processor to use a calculated value. If a manual rate is desired, the rate in counts of the 16Mhz clock per bit time (word count) must be entered. The display processor converts the user input frequency to word count and places it here if required.

The word length is obvious, the number of bits per words. Actually any value will be processed up to 16 but for word lengths of less than 8, the time between words at high data rates may not be sufficient for the system to process the data and lose sync. Values above 16 will be yield the same results as 16. The PCM processor uses this value to create a mask to mask out the unwanted bits in a data word. If, for example, the value of 8 was chosen, the upper 8 bits of the 16 bit word read from the word processor would be masked out. This is necessary because the word processor shifts in all bits and can hold two 8 bit words at a time. Looking at a whole 16 bit word without masking would yield two data words when only one was needed and would never find the sync words.

The frame length is the number of words per frame not including the synchronization words. The PCM processor uses this value to set counters which will alert the system when to look for the next sync word/s. If the wrong value is entered, synchronization will occur but for only one frame at a time as the sync words will be in a different place with respect to where the system thinks they are.

The next three addresses should contain the sync word values. The

value three was determined from past history of PCM usage within the 4950th Test Wing. The first location should contain the first value, the second is next and the last address should contain the third. A special note here to the the programmer who wishes to modify the display processor software: The polarity must be applied to the sync word definitions before sent to the I/O control block. The PCM processor reads in all data in normal format and inverts it, if necessary, if inverse POLARITY is in effect. There is enough time to do this when reading data at the word rate but not at the bit rate. If the polarity is inverse and the user inputs a sync word value of "00001111", it should be converted to "11110000" before sending it to the I/O control block.

Location $C7EE is an output location in which the PCM processor sends the error count to the display processor to be used in the status page in program EXECUTE. The error count is incremented each time synchronization has been established and then lost. If no signal were present, no errors would be reported. In other words, an error implies that the system was in sync at some time. If the system wasn't in sync, there could be no error.

Polarity is inputted to the PCM processor through location $C7EC in the I/O control block. Polarity is simply the reversal of the voltages and the bit values they represent. Normally all PCM signals represent a "0" as a low voltage level and a "1" as a high level. In Bi-Phase, a transition from low to high normally represents a "0" and a transition from high to low represents a "1". In some systems tested in the past on flight test programs, the Bi-Phase transition meanings

were reversed. The POLARITY capability was added to allow for such non-standard systems.

The frames/buffer input was added to avert additional division routines in the PCM processor software. The calculation is done in the display processor software to take advantage of the division operation of the BASIC language. The value is used to determine when the buffer is full.

The PCM processor outputs the calculated pcm type as an ASCII value in the next location, $C7E8. Whether or not the user wants to use a preset PCM type, the PCM processor does the calculation anyway and returns it to the user. The value is displayed on the status page of the display processor.

Likewise the calculated PCM rate is returned in location $C7E6. The PCM rate is calculated regardless of whether the user wishes it to be or not, and returned on the status page. This value can be used to determine if the PCM stream is at the proper bit rate or not.

The correction value is returned in location $C7E4 and is indicative of the jitter in the PCM stream. The calculate rate routine finds the minimum number of counts between two positive going pulses. For an entirely stable PCM data rate, the minimum will always be the same within one count. If jitter is present, the minimum count will be somewhat less than the actual bit rate times two. Note that the count between two positive going signal pulses is twice that which occurs during one bit time. If the bit rate is high enough and the jitter is bad enough, the walk error explained in section IV will become substantial and cause the system to break sync. The PCM processor

corrects for this by adding 1 to correction each time synchronization is lost. If the value becomes larger than 1Ø, it is reset to Ø. In this way, the system keeps on track with even the most varying PCM signal. The maximum value of 1Ø was chosen as an educated guess and laboratory test explained in section IV. The correction counter is returned in the I/O control block location correction.

The status location of the I/O control block contains a value of non-zero (true) when the PCM signal is in sync, and zero (false) when not. When the display processor finds a false value in status, updates of data values are halted.

The remaining locations in the I/O buffer are for data word values. As the system controller reads in the data words, it stores them here beginning at address $C000 and ends at the integer value of $C000 + WORD LENGTH times FRAME LENGTH. The maximum value assures that the buffer will not be overfilled.

The system controller interfaces to the rest of the PCM processor system through the remaining addresses listed in table 2. These were described fully in the hardware section and will not be repeated here. See appendices B and C for flowcharts and listings of the PCM processor software.

III. Display Processor Design

System Overview

The display processor's main function is to control and read the data from the PCM processor then display it in a format conducive to the user's desires. In order to accomplish these functions, the display processor is organized into two sections: A microcomputer with interface to the PCM processor card, and the system software.

The microcomputer was chosen to be a Radio Shack Color Computer II because of the size and capability. The interface to the PCM processor is described below in the hardware section but in essence, the PCM processor is interfaced to the display processor much like that of ordinary RAM.

The system software was written primarily in the BASIC language for ease of programming and maintenance. Since the overall operation of the display processor is performed in software, this overview will concentrate on how the system software works. Later sections describe in detail the operations of each function.

Requirements for the system call for a display in engineering units. A few added features are provided to display bar graphs and plots of selected parameters. Also, the system must be able to accept from 8 to 16 bit wide words and a PCM stream of up to several hundred data words or parameters. The incoming PCM signal can be anywhere from 1 Khz to 100 Khz. Because this system will be used to monitor data only, it will not be necessary to capture each and every data word. For a PCM signal of 16 bit words as a rate of 100 Khz, the word rate would be about 6250 words per second. Although the PCM processor can

capture words at this rate, only a sample of the data need be processed to be effectively monitored. This allows the use of a high order language like BASIC to be used for ease of programming and maintenance. A special routine has been be added to extract up to 1000 frames of one data word to search for glitches etc. This will be referred to as trace mode of operation. During normal operations of a flight test, all data words are independently recorded on tape for laboratory investigation.

The system software is designed to do five major functions.

1. Allow entry of system frame specifications.
2. Allow entry of parameter data to the system.
3. Define what data is to be displayed.
4. Display the data in various formats to the user.
5. Alert the user of system problems.

All functions are integrated into an environment which is completely menu driven with appropriate command prompts and error checking throughout. Some examples fully explained later in this section are as follows: The system won't accept erroneous data as input; the user is notified before a function abort; when conflicting data is entered such as setting an alarm beyond the range limits of the input data, the user is prompted, returned to the area where the conflict occurs, and given the opportunity to resolve the problem; and changing the number of sync words from 3 to 2 erases the 3rd sync word as it no longer applies. Many other subtle considerations such as these are scattered throughout the environment to make it a professional package which can be used quickly and easily.

System layout is shown on the data flow diagram in figure 19. The programs within the environment are all version 1.0 and are as follows:

1. MAIN.BAS
2. FSETUP.BAS
3. PSETUP.BAS
4. DSETUP.BAS
5. CONVERT.BIN
6. PARAM.DAT
7. DISPLAY.DAT

The Color Computer Disk Operating System appends the extension BAS to BASIC files and DAT to data files. The user can choose a different extension if desired.

The user begins his interaction with the program titled MAIN. This menu driven program allows the user to do one of five functions. He/she can create a new system data base by clearing the old one, enter parameter data, enter display data, enter frame specifications, or display the data set up in previous sections. There are two data files called PARAM.DAT, and DISPLAY.DAT. Each of these files are accessed by all programs as shown on the diagram. There is also a short machine language program called CONVERT.BIN which does some the data conversions done very efficiently in machine language. Such conversions are hexadecimal to binary or hexadecimal to octal.

The system works as follows: First the user clears the old data files with the "CREATE" option from MAIN. This option is actually a section of MAIN because of its simplicity. Once cleared, new data can be entered. The user must then describe the frame specifications using the "F)RAME SPECIFICATION" option. This option must be accomplished before the user uses the "X)ECUTE" option as the PCM specifications will most likely be wrong. The "F" option runs a program called FSETUP which allows the user to tell the system how many and what the sync

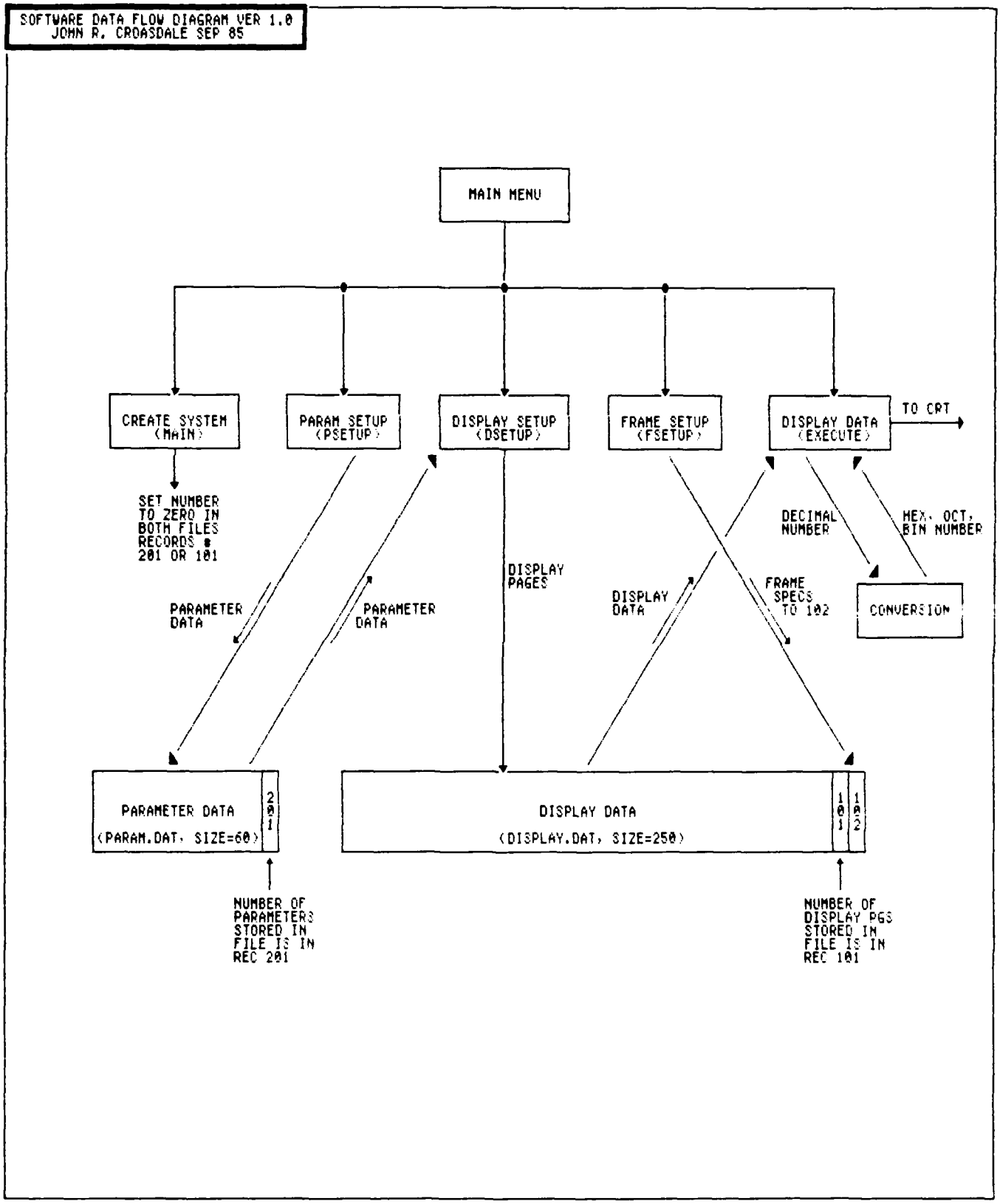


Figure 19. Display Processor Data Flow Diagram

words are in the PCM stream, the number of bits per word, the number of words per frame, the PCM type and polarity, and the bit rate. FSETUP stores the frame specification in record 102 of DISPLAY.DAT to avoid unnecessary data files. See appendix A for a listing of the file spec protocol.

The next thing is to define the parameters or data words in detail using the "P)ARAMETER SPECIFICATIONS" option from MAIN. This option calls the program called PSETUP. The system can hold up to 200 parameters at one time.

Once the data word data base has been defined the user must arrange the previously entered data into a display page format using the "D)ISPLAY FORMATS" option. This option runs the program DSETUP. Each page can consist of up to 10 parameters. Only one page can be seen at one time but up to 50 pages can be accessed for prompt display. By defining page formats in advance, execution speed during display time is greatly enhanced because most of the disk access work is done.

In summary, FSETUP specifies the frame definition, PSETUP fills PARAM.DAT. DSETUP then uses data from PARAM.DAT to create files in DISPLAY.DAT. EXECUTE uses this data along with the frame specification in record #102 to display the appropriate data to the user. More explicit explanation follows.

Color Computer Interface

The design of the display processor hardware interface to the PCM processor is very simple due to the dual ported RAM explained earlier. The interface schematic is shown in figure 20. The PCM processor

appears to the Color Computer's I/O port like 2048 bytes of RAM. The interface maps the RAM at locations from $E000 to $E7FF. To make the circuit even simpler, the RAM is also mapped into locations $E800 to $EFFF. By doing this, only the four highest address bits need to be decoded. Whenever address bits A15, A14, A13, and A12 are at states 1, 1, 1, 0, respectively, the output side of the I/O RAM will be selected by the display processor.

Because the PCM processor is a 16 bit system, it writes to the dual port RAM in 16 bit chunks. The MC68000 microprocessor, however, addresses only bytes so the address range is the same to both the PCM processor and display processor (2k bytes). Only the locations are different. If the MC68000 microprocessor writes a word to location $C000 with respect to the PCM processor (see section on the System Controller), it will write the high 8 bits to $C000 and the low 8 bits to $C001. This data will be presented to the display processor at locations $E000 and $E001 respectively. Location $E000 contains the high order byte while $E001 contains the low order byte. This applies to every even and odd address throughout the dual port memory.

The interface decodes the address range $E000 to $EFFF, and enable either the low byte with an odd address (A00 is a 1), or an even address (A00 is a 0). This is done by first decoding the four high address bits, A12 - A15 with the NAND gate AB6 and AND gate AC6. When $Exxx is addressed (x means don't care), pin 6 of AC6 will go high and allow the other sections of AB6 to pass address bit A00 through. When this occurs, A00 enables either the even byte, or odd byte of the dual port RAM depending on whether it is at a logic level 1 or 0. For even

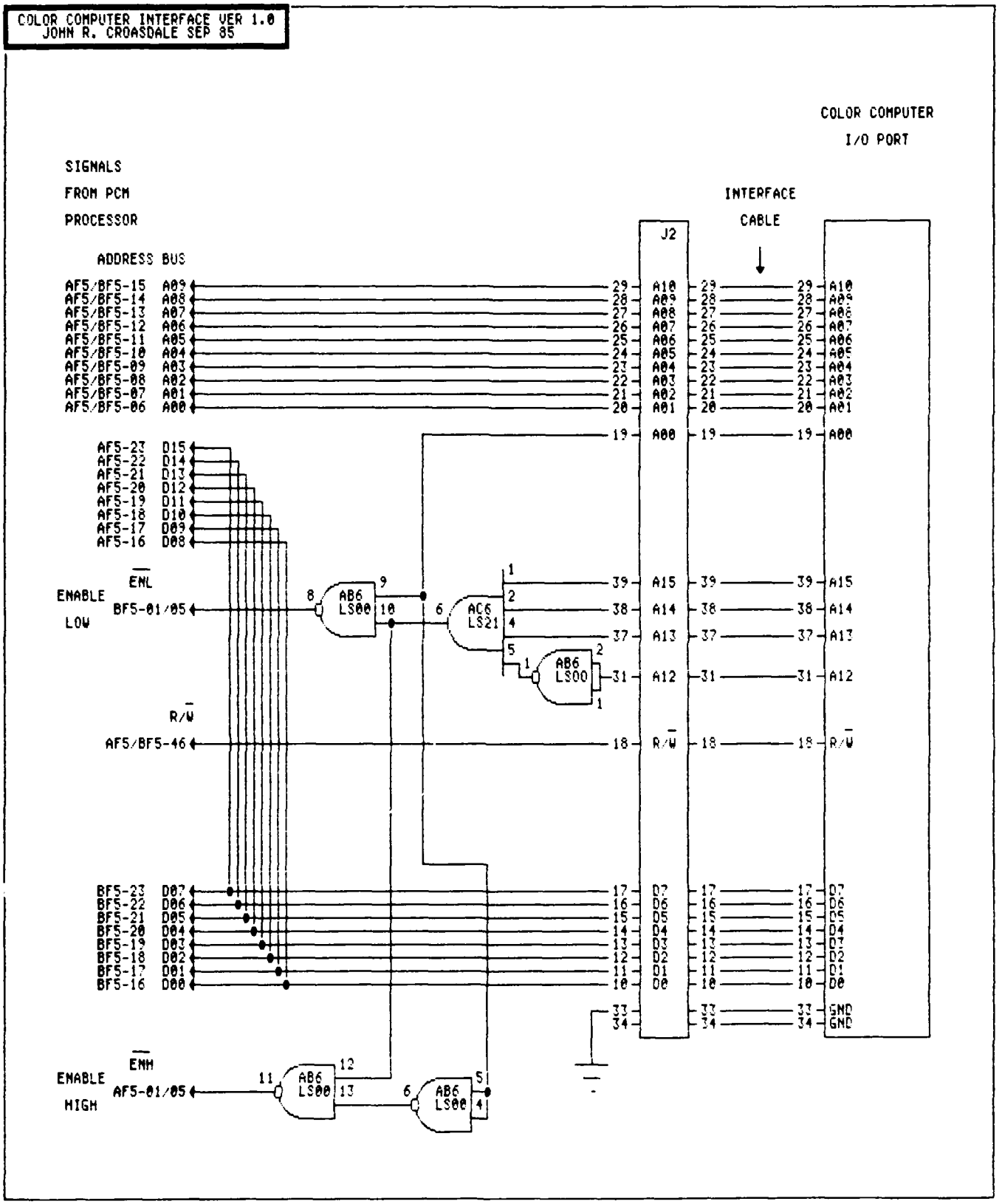


Figure 20. Color Computer Interface

addresses, A00 is low causing pin 8 of AB6 to go high, deselecting the low I/O RAM chip. It is also negated by one section of AB6 and applied to pin 13 of AB6. Since both pins 12 and 13 are now at a high logic level, pin 11 goes low enabling the high I/O RAM chip.

When the processor addresses an odd byte, A00 is now at a logic level 1 which performs the reverse operation to the I/O RAM chips.

The interface is built on the PCM processor card to avoid duplication of effort. A cable linking the two systems is connected to J2 on the processor card. This cable must be kept as short as possible (less than one foot) to avoid interfering with the Color Computer's system bus. To avoid confusion, the pin numbers for J2 are kept the same as those in the Color Computer.

In summary, the display processor will read data in the same order as the PCM processor writes it. That is, even bytes will contain the high order bits of the data, and the odd addresses will contain the low order bits. To obtain the full data word value, the display processor must multiply the high order byte by 256 and add it to the low order byte. This will yield the full value of the 16 bit word.

MAIN

Program MAIN's function is to provide as an interface between the other modules of the system. See figure 21 for the MAIN master menu. The menus shown here and throughout this chapter are actual page formats from the display processor system. Since the Color Computer presents data in 32 columns by 16 rows max, each page was designed to fit within this limitation.

MAIN calls other programs from this menu and gets recalled when the quit option is selected from them. It acts as the arbiter for the system. MAIN does have another function, however. and this function is to setup new data bases. See figure 22 for the setup menu. Only one data base at a time is allowed under the current configuration. This data base is contained in two files named PARAM.DAT and DISPLAY.DAT. The user is responsible for saving old files under a new name if they are to be kept for later use. Setting the number of records stored in each file to 0, makes them ready to accept new data. The current number of records stored in PARAM.DAT is placed in it's 201'st record. Likewise, DISPLAY.DAT's 101st record contains the number of display pages entered in the system. The data entry programs, PSETUP and DSETUP, keep these records updated at all times because many of the sub-programs in the system check the number of records stored and use the information to alert the user if access of non-existent records is attempted.

Program MAIN alerts the user when the "S)ETUP NEW SYSTEM" option is selected that the data base will be erased if he/she chooses to continue.

FSETUP

The first thing a user should do after clearing the data files is to describe the frame specification using the "FRAME SPECIFICATION" option from MAIN. Calling this option brings up the FSETUP master menu shown in figure 23 below.

As in the program MAIN above, selections are made by typing the

```
**********************************
*                                *
*           MAIN MENU            *
*                                *
**********************************

  F)RAME SPECIFICATION
  P)ARAMETER SPECIFICATIONS
  D)ISPLAY FORMATS
  X)ECUTE TO BEGIN PCM MONITOR
  S)ETUP NEW SYSTEM
  Q)UIT

   WHICH ONE-->?
```

Figure 21, MAIN MASTER MENU

- -

```
**********************************
*                                *
*       SETUP NEW SYSTEM         *
*                                *
**********************************

WARNING WARNING  WARNING WARNING

THIS PROGRAM WILL DELETE YOUR
DISPLAY FILES.  BE SURE YOU'VE
SAVED THEM TO ANOTHER FILENAME

DO YOU WISH TO CONTINUE OR ABORT
PRESS C OR A ->?
```

Program MAIN has a master menu page shown in the top of this figure. If the S function is called, the lower page will be displayed. If the user responds with a "C", the program will set up a new system data base. Any other key will revert back to the main menu.

Figure 22, SETUP MENU

first letter in the line before the parenthesis followed by the "ENTER" key. The FSETUP program gives the user the options of creating a new frame specification, editing a specification in memory created during this session, loading the frame specification saved on disk from a previous session, or quitting. The system can hold only one frame specification at a time. It stores the specification in the DISPLAY.DAT data file in record #102. Appendix A describes the disk format for the frame specification.

The first option, "C)REATE", brings up the data entry page shown in figure 24. There are seven items which need to entered to describe the frame format. The arrow keys are used to move up and down the seven line entries and to scroll left and right within a line. The first line is "BITS/WORD". Here the user types a number up to 16 bits. The normal size of a PCM word is around 10 to 12 bits but for greater resolution, 16 bits can be used. The number of bits, as well as all the other entries on this page, is dictated by the PCM encoder and not the PCM monitor. The user must use what is in the PCM stream.

The second line entry is "WORDS/FRAME". Any number of words are allowed up to a total of 200. "PCM/TYPE" is either NRZ-L or bi-phase-L. These types are explained in the introduction. A third option on this line is "AUTO". The PCM monitor has the capability of determining whether the signal is in NRZ format or bi-phase. Selecting "AUTO" allows the system to use the calculated type.

Perhaps the most important innovation of the PCM processor is its capability to capture a PCM bit rate from 1Khz to 100Khz automatically. The user has the option of manually setting a bit rate by typing a

```
**********************************
*   FRAME SPECIFICATION PAGE     *
**********************************

C)REATE NEW FRAME SPECIFICATION
E)DIT SPECIFICATION IN MEMORY
L)OAD FRAME SPECIFICATION
Q)UIT

WHICH ONE->?
```

Figure 23, FSETUP MASTER MENU

```
**********************************
*   FRAME SPECIFICATION PAGE     *
**********************************

BITS/WORD       10  MAX IS 16
WORDS/FRAME     34  MAX IS 200
PCM TYPE        AUTO NRZ BI-PHASE
PCM BIT RATE    AUTO KHZ A)UTO
PCM ORDER       LSB MSB
PCM POLARITY    NORMAL INVERSE
# OF SYNC WORDS  3  MAX # IS 3
SYNC DEF->  1111101011
            1100110011
            0100000000
**********************************
 USE NUMBERS, ARROWS, OR CLEAR
```

Figure 24, FRAME SPECIFICATION PAGE

number in the PCM bit rate line. Such a number could be 25.5 meaning 25.5 thousand bits per second. By typing the letter A, the "AUTO" feature is selected. If a number is used, the PCM processor will use this number but it will return the calculated value to the user. The calculated value will be displayed on the status page of the EXECUTE program explained later.

PCM order can be either Most Significant Bit (MSB) first, or Least Significant Bit (LSB) first. The is determined by the PCM encoder. Looking at a digital word of (10110100), for example, the MSB is on the left and is a "1". The LSB is on the left "0". The MSB has the highest weight or value while the LSB has the lowest. Generally, PCM systems send data LSB first. If the PCM monitor is not told what this order is, the data may become transposed and be meaningless. The word above has a value of 180 decimal. If it were transposed to (00101101), its value would be 45. The order is switch selectible on the PCM processor card.

The PCM polarity option is used to switch from a high level meaning "1" to a high level meaning "0" etc. It is normally used for bi-phase as some systems use a transition from low to high to mean "0" and high to low to mean "1". Other systems use the reverse. The polarity function is accomplished in the PCM processor software to allow for any type of encoder polarity outputs. Consult the software section of the PCM processor for more information on polarity.

The number of synchronization words is entered on the next line. The word length for the sync word must be the same as the normal word length. Up to 3 sync words can be used. The "SYNC DEFINITION" lines

follow to allow the user to enter the binary representation of the word. If only one sync word is allowed, the program will allow only one to be entered. If two are allowed, only two can be entered. The same for three. The program also checks the word size from line one and allows the user to enter only those number of bits. If the BITS/WORD is changed for some reason, the "SYNC DEFINITION" display area is erased and must be re-defined. This keeps the user from entering erroneous data

Pressing the "CLEAR" key at any time during the data entry process saves the data displayed shown on the page to disk. If items are left blank, the default options shown in inverse are used.

The next option from the master menu is "E)DIT". This allows the user to re-enter the frame specification page without re-loading from disk. The "L)OAD" option brings in the saved file from disk and enters the specification page. "Q)UIT" returns the user to the MAIN program.

PSETUP

The parameter setup program, PSETUP, allows the user to set up parameter file, PARAM.DAT. Parameters are those items which are to be measured and used to determine if the system under test is functioning. A parameter, for example, might be motor temperature for the left outboard flap motor. A parameter could also be the cabin altitude aboard an aircraft. Because not all parameters need to be placed into a PCM data stream, the parameter number and data word number do not necessarily need to be the same. When the user puts a parameter list together, he decides which parameters need to go into the PCM system.

PSETUP therefore allows the user to have different parameter numbers than data word numbers. Data word numbers depict the position in the PCM stream where the parameter is located. Parameter number 15 might be the 2nd word in the data stream for example.

When called from the MAIN program, PSETUP displays the master menu page depicted in figure 25 below. The user is offered the options to "A)DD" new words, "E)DIT" words already in the system, "I)NSERT" data words between others, "D)ELETE" data words, "L)IST" all the data entered in the system, or "Q)UIT".

The "A)DD" selection brings up the page shown in figure 26. All the data needed by the system is shown on this data entry page. By adding a new data word, the system looks up how many have been entered before and adds one to it. It then displays this number in the top line of the page.

The next entry the user must provide is a parameter name. Because of the limited number of columns available on the Color Computer, only seven characters are allowed for the parameter name. This number was arrived at because of other page limits in the PCM Monitor system. The figure shows a parameter name of "ALTITUD" (not a typo) for aircraft altitude. Its position on the users parameter list is four so its parameter number is four but it is the first word in the data stream. By keeping track of the parameter number, the user can relate data shown on the PCM Monitor with the master list. Parameter units are obvious, feet.

The range minimum and range maximum relate to the full range of the expected output of the transducer supplying the data to the PCM

```
*********************************
*                               *
*   DATA WORD DEFINITION PAGE   *
*                               *
*********************************

A)DD NEW DATA WORD
E)DIT DATA WORD
I)NSERT DATA WORD
D)ELETE DATA WORD
L)IST PARAMETER TABLE
Q)UIT

WHICH ONE?
```

Figure 25, PSETUP MASTER MENU

```
   DATA WORD DEFINITION PAGE
*********************************
ADDING DATA WORD # 1

PARAMETER NAME --> ALTITUD
PARAMETER NUMBER-> 4
PARAMETER UNITS -> FEET

RANGE MINIMUM -> -100
RANGE MAXIMUM -> 50000
ALARM MINIMUM -> 200
ALARM MAXIMUM -> 45000

*********************************
ENTER FOR NEXT, CLEAR TO ACCEPT
```

Figure 26, A)DD NEW DATA PAGE

encoder. For example, when the digital data for this data word is zero, (all bits are 0), the user should expect to see a certain value on the display which is set up during the transducer calibration. In this case, a zero reading would be scaled to -100 feet. If the data words were 16 bits long, the maximum absolute data read into the system would be 65535 or all bits set high (1). The system would then be calibrated to give a maximum reading of, in this case, 50000 feet. In no way could the system display values lesser or greater than range minimum or range maximum respectively. The absolute readings from the 16 data bits in this example would then be mapped or scaled to fall within the range high and range low limits and be presented to the user. This process is described in the EXECUTE section.

Alarms are set to alert the user if the data is exceeding a preset value. In this case, the user wants to know when he/she is getting close to the ground so alarm low was set to 100 feet. In addition, a high alarm was set to 45000 feet to alert the user when the aircraft is above a save limit. Alarms are displayed only on the bar chart display explained in the EXECUTE section. These useful prompts can tell the user of imminent brake damage due to excessive temperature, or when the cabin altitude is dangerously high etc. Very few systems units on the market have this feature in their PCM display units.

At the bottom of the page are several command prompts. The "ENTER" keys on the Color Computer is used to jump from one line to the next. In addition, the arrow keys are also used to move within a line, or from one line to another. Pressing the "CLEAR" key brings up the save page shown below in figure 27.

The user can either "SAVE" the data just entered to disk and return to the master menu, "TRY" again to change data just entered, or "RETURN" to the master menu without saving the data. Data is saved in the PARAM.DAT file in a format explained in appendix A.

Error checking is done throughout to prevent the user from entering data which would bomb the system at a later time. For example, if an alarm were set outside the range limits, the user would be alerted and returned to the alarm entry position on the add data word page. Alarms out of limits would cause the plot function, for example, in the EXECUTE program to plot off the screen. Trying to specify a parameter name as a number, however, is allowed as the system doesn't care what the user calls his/her parameters. Alpha characters such as "A" or "D" are accepted as a zero value in cases when a number is expected to add flexibility.

The "E)DIT DATA WORD" option allows the user to edit a previously

```
S)AVE DATA WORD # 1
T)RY AGAIN
R)ETURN TO MENU

WHICH ONE ->?
```

Figure 27, Save Page

defined parameter. When this option is selected, the user is asked which data word he/she wishes to edit. At this point, a number will bring in the specified data word to a page exactly like the add page described above. The user has several options other than editing a data word. These options are EXIT and List. "EXIT" returns the user to the master menu. "LIST" presents a display shown in figure 29 which is a summary of all the parameters entered in the system. This "LIST" command is exactly the same as the "L)IST" function on the master menu with the exception that pressing a number will bring up that data word into the edit data word page. If the "LIST" function is brought up from the master menu, pressing a number will only return the user to the master menu as the system does not know what to do with it, edit, delete, insert or whatever?.

Listing a data base helps the user get his/her bearings if the data word number for a particular parameter can't be remembered. It is also very useful in checking to see if all parameters are entered correctly. Of course, the characteristics of the parameters are not displayed but at least its position in the data stream can be verified.

The user has three options from the listing page, all of which are effective when called from the "EDIT" option. Pressing a number will bring the selected parameter into the edit parameter part of the program. The user can edit at will and save as mentioned above. A carriage return, or "ENTER" key on the Color Computer, will continue the listing if more than ten data words are entered into the system. The listing displays only ten lines at a time to allow the user time to see them. Pressing the "E" key exits the listing and returns the user

```
ENTER DATA WORD TO EDIT ->?


PRESS L TO LIST, E TO EXIT
```

Figure 28, Command prompts from EDIT function

- -

```
DATA WORD LISTING OF 12  RECORDS
 *********************************
      WORD #       NAME  PARM

        1         ALTITUD  4
        2         AIRSPED  5
        3         CAB ALT  6
        4         CAB TEM  7
        5         BRK1 TP  10
        6         BRK2 TP  11
        7         BRK3 TP  12
        8         BRK4 TP  13
        9         AIR PRS  19
        10        OAT      25

NUMBER=SEL, CR=CONT, E=EXIT ?
```

Figure 29, LISTING DISPLAY

to the master menu.

The "I)NSERT DATA WORD" option allows the user to insert data word specifications between other data words. This feature is very useful if changes to the parameter list or sequence of the data word order have been made for some reason. Selecting this function brings up a user prompt page similar to that which was brought up if the "E)DIT DATA WORD" were selected. The user has the same options, "ENTER" a data word number, "LIST", or "EXIT". If a data word number is entered, the insert data word page will be displayed. This page is exactly the same as the add data word page except for the title. The new data word will be inserted before the number selected by the user. The user can abort at any time but if he/she chooses the "SAVE" option, the data base will be re-ordered to reflect the inserted word. This operation is entirely disk based so for large data bases, the time to complete the operation could take up to 15 minutes. For this reason, the user is continually prompted of the disk operations performed during the operation. He/she cannot abort once the operation is underway.

The "D)ELETE DATA WORD" option allows the user to delete a data word. The file is then re-arranged to assure sequential data word numbers. This feature is exactly the same as the "I)NSERT DATA WORD" option except that the user is shown the data word he/she has chosen in detail on the same page format as the add or edit data word page. At this time, the user is re-asked if it is to be deleted. If yes is chosen, the file is re-arranged in the same process as the "INSERT" option.

The "LIST" option does exactly the same function as the "L)IST"

option in in the edit, insert, and delete modules. There is one exception and that is if a data word is selected from the listing, the program returns to the master menu instead of displaying the selected word. This option is intended to be used to double check all data entries before moving on to the "D)ISPLAY FORMAT" option from the MAIN program.

The final option of PSETUP is "Q)UIT". Calling this function returns the user to the MAIN module and allows him to call other programs in the system.

DSETUP

The fourth option in the MAIN program is "D)ISPLAY FORMATS". This option runs the DSETUP program and allows the user to set up the display page data. The formats for the display are built into the system in the EXECUTE program explained below. DSETUP allows the user to specify which parameters or data words he/she wishes to see at one time. Up to 10 words can be inserted into one display page. These 10 can be any combination of the data words defined using the previous PSETUP program. The user can choose to make a page of just one data word iterated 10 times to allow for a higher sampling rate in program EXECUTE. Another example would be to show the data for a particular parameter scaled to the range limits along with its raw value from the PCM processor is displayed. These options are available as explained later in the USERS MANUAL.

The DSETUP master menu is shown in figure 30. The user is presented with seven options described below. The first option is

```
*********************************
*   DISPLAY DEFINITION PAGE     *
*********************************

A)DD NEW DISPLAY PAGE
E)DIT DISPLAY PAGE
I)NSERT DISPLAY PAGE
D)ELETE DISPLAY PAGE
L)IST CURRENT DISPLAY PAGES
P)ARAMETER LISTING
Q)UIT

WHICH ONE->?
```

Figure 30, DSETUP MASTER MENU

```
      ADDING DISPLAY PAGE # 2
*********************************
  POS   DW#   PAR    NAME     TYP
   1     1     4     ALTITUD  DEC
   2     1     4     ALTITUD  HEX
   3     1     4     ALTITUD  OCT
   4     1     4     ALTITUD  BIN
   5     1     4     ALTITUD  SWI
   6     2     5     AIRSPED  DEC
   7     2     5     AIRSPED  DEC
   8     3     6     CAB ALT  DEC
   9     4     7     CAB TEM  DEC
  10     10    25    OAT      DEC
*********************************
 ENTER=NEXT CLEAR=ACCEPT SP=TOG
```

Figure 31, ADD/EDIT DISPLAY PAGE

"A)DD NEW DISPLAY PAGE". Selecting this option will present the page shown in figure 31.

The same scrolling technique is used in this page as in all the others before. The arrow keys move the cursor from left to right. The enter key scrolls the user to the next field for data entry. Only two fields per line are allowed. By typing a number in the first field of the line followed by the "ENTER" key, that data word will be brought into memory from disk and its parameter number and name will be displayed. The next field is asking the user which data type it is to be displayed in. Five data types are allowed and toggled through using the space key. The types are DECimal, HEXadecimal, OCTal, BINary, and SWItch. The first four are self explanatory but the SWItch is a new feature. This simply tells the user if the data is non-zero, or zero. If it is zero, the EXECUTE program will show the data as "OFF". If it is non-zero, it is "ON". It is designed to be used as event markers for discrete actions such as whether a system is turned on or is receiving a signal. Pressing the space bar will toggle through the five types and when the user sees what is desired, the "ENTER" key accepts the data and scrolls to the first field on the next line. The example shows that the user wants to see "ALTITUD" in all five different data types, "AIRSPD" in two sequential frames, and a sampling of CABin ALTitude, CABin TEMperature, and Outside Air Temperature. All this data will be displayed as page one in program EXECUTE. When the user is finished describing the page, the "CLEAR" key is pressed to "SAVE" the data, "TRY" again to edit the data, or "RETURN" to the master menu without saving. This is the same process used in the

PSETUP program for saving parameter specifications.

The "E)DIT" option is handled much like the "E)DIT" option from the PSETUP program. The user is prompted for a page number and has the choice of entering one, LISTing the display page file, or EXITing back to the master menu. The prompts are not shown here but are the same as those found in PSETUP and work the same way. If a number is selected, that page will be displayed in a format exactly like the "A)DD" page except that the header will show that the user is editing rather than adding.

Selecting "LIST" will bring up the display listing page shown in figure 32. Because of the limited column capability of the Color Computer, the listing is rather busy. Along the top, the listing shows how many pages are defined in the system. In this case there are 12 pages, the first 10 are shown in the listing. If the "ENTER" key is pressed, the last two pages will be presented as shown in figure 33. The line between the asterisks shows the position number of the 10 entries per page. The left column shows the page number with the data word numbers on that page across the line. This format allows the user to see what data words will be displayed when a particular page is displayed in EXECUTE. The user can bring a page into EDIT by simply typing its number so long as it is shown on the catalog page. Pressing "ENTER" will bring up the next 10 pages, if there are 10 more. This cycling will continue at the users prompt until the entire catalog is displayed. The user can elect to quit using the "EXIT" option. When entered from the "EDIT" option on the master menu page, the catalog will return back to the "EDIT" option if "EXIT"ed. This is the same

```
  CATALOG LISTING OF 12 PAGES
**********************************
PG!1!!2!!3!!4!!5!!6!!7!!8!!9!!10
**********************************
1 1   1   1   1   1   2   2   3   4   10
2 11  12  13  14  15  16  17  18  19  20
3 21  22  23  24  25  26  27  28  29  30
4 31  31  31  31  31  31  31  31  31  31
5 32  32  32  32  32  33  33  33  33  33
6 34  35  36  37  38  39  40  41  42  43
7 11  15  17  19  23  35  43  44  45  46
8 9   9   9   9   9   9   9   9   9   9
9 50  51  52  53  54  55  56  57  58  59
1060  61  62  63  64  65  66  67  68  69
**********************************
NUMBER=SEL, ENTER=CONT, E=EXIT ?
```

Figure 32, DISPLAY PAGE LISTING

- -

```
  CATALOG LISTING OF 12 PAGES
**********************************
PG!1!!2!!3!!4!!5!!6!!7!!8!!9!!10
**********************************
111   1   1   1   1   2   2   3   4   10
1211  12  13  14  15  16  17  18  19  20
```

```
**********************************
NUMBER=SEL, ENTER=CONT, E=EXIT ?
```

Figure 33, DISPLAY PAGE LISTING cont

methodology as that used for the previous listing pages in PSETUP and in the other options in DSETUP. The calling routine will be returned to if "EXIT" is used from the catalog.

The "I)NSERT" option from DSETUP's master menu will perform the insert utility similar to that is PSETUP. The user will be asked where to insert the display page, "LIST" the catalog, or "EXIT". By using the list function the user can see where he wants to insert a page. Typing a page number will bring up a blank DISPLAY PAGE for the user to fill in. The page will be inserted in the DISPLAY.DAT data file before the number page number selected and assume its value. Like the "INSERT" function in PSETUP, inserting a page can be quite a lengthy process so it must be used wisely.

The "D)ELETE" option will delete a display page in the same fashion as the "INSERT" option except the selected page to delete will be presented to the user for a final go ahead. At this time he/she can delete the page or abort the operation.

The "L)IST" option from master menu will display a catalog listing exactly the same as used before but will return to the master menu if a page is selected as the program will not know whether to edit, insert, or delete the page.

The "P)ARAMETER LISTING" option will list values of the selected data word exactly the same as in the PSETUP program. Consult the section on listing parameters for more information on this option.

The "Q)uit" option exits the program and returns to the MAIN programs master menu.

EXECUTE

The EXECUTE program is the workhorse of the system. It is the program responsible for displaying the data set up in PSETUP and DSETUP and presenting it in user readable formats. It also performs the function of providing system status feedback to the user. Calling the "X)ECUTE" option from MAIN will run the EXECUTE program. EXECUTE will prompt the user to set the PCM order switch on the PCM processor card to either MSB or LSB depending on what was input to the system during the frame specification setup procedure. After a few seconds, the EXECUTE master menu will be displayed. This menu is shown in figure 34.

Calling the "S)TATUS" option brings up the status display page shown in figure 35. This page is for display only, no input is possible. Pressing any key returns the user to the master menu. Much of the display is the frame specification entered earlier. The two columns, "USER" and "CALCULATED" display the data entered by the user, and calculated by the system respectively. Where items are not calculated, the display is filled with dashes. By looking at the sample system, it can be seen that the bits/word and words/frame were user entered during the frame specification procedure. The user selected auto PCM type calculation and the system calculated NRZ. The user wanted to manually enter a bit rate of 25.5 Khz but the system calculated 25.9. Since the system is running off a separate clock which may or may not be as accurate as the encoder system, the calculated rate may not be exact. The important thing is that the system is tuned to the calculated rate since it uses the incoming PCM

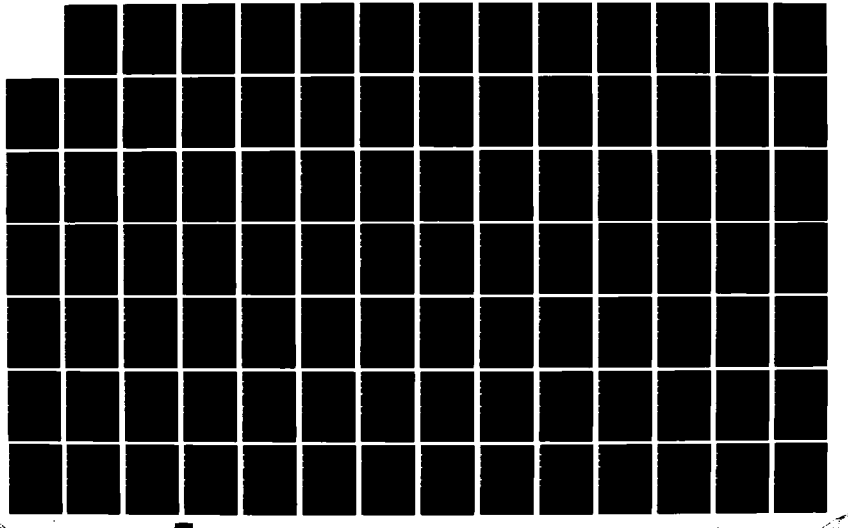
```
**********************************
*      PCM DISPLAY MODULE        *
**********************************

S)TATUS PAGE
B)EGIN DISPLAY PROCESSING
Q)UIT

WHICH ONE->?
```

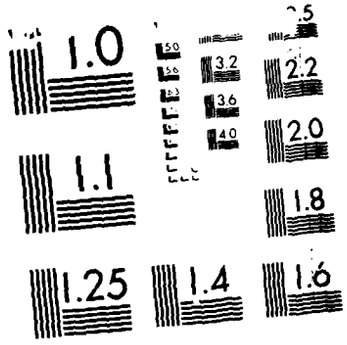
Figure 34, EXECUTE MASTER MENU

```
**********************************
*   STATUS PAGE, KEY TO EXIT     *
**********************************
                  USER  CALCULATED
BITS/WORD          10       -----
WORDS/FRAME        30       -----
PCM TYPE          AUTO       NRZ
BIT RATE          25.5       25.9
PCM ORDER         MSB       -----
PCM POLARITY      NORMAL    -----
SYNC ERRORS       -----      10
PCM HEALTH        -----      3
SYNC WORDS 1- 1111101011
   INVERSE 2- 1100110011
IF IN SYNC 3- 0100000000
```

Figure 35, STATUS DISPLAY PAGE

AD-A164 835 REAL-TIME FLIGHT TEST PCM DATA ACQUISITION MONITOR(U) 2/3
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
OF ENGINEERING J R CROASDALE SEP 85 AFIT/GE/ENG/85S-1
UNCLASSIFIED F/G 17/2 NL





signal to synchronize on. It knows what the best rate to use is and that is 25.9 Khz. The system will, however, use what the user wants, 25.5 Khz but the system may not lock on to the signal as well as it should. The main reason for allowing manual entry is if the PCM signal is erratic such as that received from a recorder or telemetry signal. In this case, the calculated rate may be too low for an accurate lock. The system iterates to zero in on the signal but it does have its limits. See the section on the PCM processor software.

PCM order and polarity are user set and not calculated. Sync errors are generated once sync is established and then broken. If no signal is present, then no sync errors will occur since the signal was never synchronized. If however, a good signal was obtained, the error counter in the system is set and incremented each time a break lock is encountered. The only way to reset it is to exit the EXECUTE program and re-enter.

PCM health is a generic term which tells the user how stable the signal is. This calculation is explained in the PCM processor software section but results in a number from Ø to 1Ø with Ø being a good stable signal and 1Ø being rather erratic. An absolute value cannot be obtained from this except that it, along with the sync error counter above, tells the user if the signal is erratic or stable. The health calculation is actually the number of counts that have to be added to the minimum count to keep the signal synchronized.

The last three lines on the status page are for the sync words. They are repeated here for user information. The word "SYNC" in "SYNC WORDS" will be written in inverse video if the PCM stream is

synchronized. If it isn't, it will be displayed in normal video. The difference between inverse and normal depends on the type of display being used. Most video monitors display white characters on a black background to inverse is black characters on a white background. If a television set is used, the display is black on a white background so inverse is white on black. In all cases, the word "SYNC" will be displayed opposite from all other words on the screen if the signal is in sync.

After returning the master menu, the only other option remaining except quitting, is to display the data. Selecting the "B)EGIN DISPLAY PROCESSING" option brings up the engineering display page shown in figure 36.

When the engineering data display page appears, the user is asked to input a page number. The user must enter a number at this point and press "ENTER". This will bring up the selected display page. The

```
   ENGINEERING DISPLAY PAGE 1
**********************************
DW#   NAME     VALUE      UNITS TY
**********************************
1     ALTITUD  13429.41   FEET   D
1     ALTITUD  0276       HEX    H <-
1     ALTITUD  1166       OCT    O
1     ALTITUD  1001110110        B
1     ALTITUD  ON         SWI    S
2     AIRSPED  250.4      KNOTS  D
2     AIRSPED  250.6      KNOTS  D
3     CAB ALT  8018       FEET   D
4     CAB TEM  78         DEG F  D
10    OAT      -22.5      DEG C  D
**********************************
<> NUM, T)YP, B)AR, P)LT, E)XIT
```

Figure 36, ENGINEERING DISPLAY PAGE

sample shows the page defined earlier using DSETUP. At this point, the data is being displayed and continually updated as fast as the display processor program can run. The data is updated from top to bottom and is converted on-the-fly using whatever conversion was selected. The conversion selected for each data word is shown in the far right column.

When the display line is set for decimal, the data is scaled and offset to engineering data reflecting the minimum and maximum range described during the parameter setup procedure in PSETUP. At any other setting, such as HEX for example, the data is presented in raw data as it is read in from the PCM processor. The units column shows the selected conversion type if other then decimal to alert the user that the data is not presented in engineering units. A mistake would be made if the user thought that 1166 OCTAL was the actual altitude in feet. For a 10 bit word, $3FF would be the maximum absolute value read in to the system. It would them be ranged and scaled to read 50000 feet, the maximum range set up in PSETUP for "ALTITUD". In this example, the raw data $276 which is converted to 13429.41 feet using simple linear interpolation techniques. Defining ABSMAX as the maximum absolute value of the data word in raw format, the scale would be (RANGE MAXIMUM - RANGE MINIMUM)/(ABSMAX). The value displayed would then be SCALE * raw value + RANGE MINIMUM. This value would be displayed in decimal format only.

The user has the option of changing the displayed conversion type at will by pressing the "T)YP" function shown at the bottom of the page. Just as described in DSETUP, the data types will toggle for the

selected line. The selected line is shown by the inverse character in the far right column under the heading "TY" for type. The example shows this as an arrow because typewriters can't type in inverse. The type will change from "D" to "H", "O", "B", and "S", and then revert back to "D". All the time, the program continues to update the data values in whatever type is selected. To change the selected line, use the arrow keys to scroll the inverse select symbol.

A special feature of the program EXECUTE is that duplicating data word numbers using different data types will show the same value under different conversion types. The first five lines shows "ALTITUD" in five different formats. Another feature is that by duplicating data word numbers using the same data type, such as "AIR SPD" in the example, the display will show two successive frames of the same word. This will only occur if the two data words are sequential on the page without another parameter inserted in between. The program checks to see if the previous data word displayed is the same as the current. If it is and the type is the same, it skips a frame and continues. If the type is different, it uses the same raw data for presentation.

Pressing the right or left arrow brings in the next or previous page for display. Pressing a number puts the user in the page number entry field at the top of the page so he/she can manually select a page for display. During the process of bringing in new pages, data update is temporarily suspended for obvious reasons.

Two other very important functions of the EXECUTE program are the "B)AR" graph and "P)LT" (plot) functions. Selecting the "B)AR" function displays the page in bar graph format. See Figure 37.

The top line of the display shows which page is being displayed. In this example it is display page 1 consisting of the same parameters shown on engineering page 1. In fact, whatever page is shown on the engineering display page will be shown on the bar graph page. The only way to get to the bar graph is through the engineering page. This holds true for the plot page shown next. The vertical bars are the alarm limits in percent of full scale which were set up during the parameter setup procedure. AT the left side of each bar is a mark which shows a rough value in percent of the data word number shown at the bottom of the page. Notice that the value for data word #3 is above the alarm limits. In this case, the user can elect to monitor only that data word using the P)LOT function described below. Above the bar at the top of the page is the actual percent value of the data word.

After any key is pressed, the user is returned to the engineering display page. From here, he/she can elect to see a plot of the selected value. The selected value is shown in inverse at the right hand column of the page, the "TY" column. Selecting "P)LT" puts the user on the page shown in figure 38. This page is similar to the bar graph page however only one data word is plotted at a time. Each point represents one frame. The intent of this feature is to look at a particular data word in detail for intermittent spikes etc. The sample rate, or time between samples as it is used here, is a function of the data rate and words per frame and is shown at the bottom of the page in milliseconds.

There are two modes of operation in "P)LT": "C)ONT" for

```
  BAR GRAPH DISPLAY FOR PAGE 1
**********************************
%  65 65 65 65 65 69 22 32 65 12
90*                  *
80* *  *  *  *  *  *  * *   *
70* *  *  *  *  * ** **  * **  *
60*** ** ** ** **  *  *  *  *  *
50* *  *  *  *  *  *  *  *  *  *
40* *  *  *  *  *  *  *  *  *  *
30* *  *  *  *  *  *     *  *  *
20* *  *  *  *  *  *     *  *  *
10*                  *     *  * **
 0*                  *
  *********************************
   1  1  1  1  1  2  2  3  4  1
DW#                              0
```

Figure 37, BAR GRAPH DISPLAY

```
 PLOT FOR DATA WORD # 1 ALTITUD
**********************************
%
90*
80*
70*
60*
50*    ***                *
40*****   ******  ******* *    *
30*             **         ****
20*
10*
 0*
  *********************************
TIME= 305.14MS    C)ONT, T)RACE
E)XIT, <> TO SCROLL, PG 1  OF 34
```

Figure 38, PLOT DISPLAY PAGE

continuous, and "T)RACE". In "C)ONT" mode, the plot routine scans each frame in the data buffer for the particular data word being plotted. It keeps updating the value in a continual basis until halted by the user. In this way, the plot is kept updated with the latest data input to the system. Because the display processor is slow compared to the PCM processor due to the BASIC language, a particular frame could be updated twice before the display processor displays it and data will be lost. To solve this problem, the "T)RACE" feature was added.

"T)RACE" commands the PCM processor to fill the memory buffer with one and only one data word. When the buffer is full, it stops and waits for the user to begin another trace. Up to 1000 data words can be stored in the memory buffer at one time so up to 1000 frames will be captured. Each frame will be continuous with no data loss. The user can scan the buffer 29 frames at a time by using the left and right arrows. This equates to 1000/29 or 34 pages of information. By pressing "C)ONT", the user will be returned to the "C)ONT" function. By pressing "T)RACE", a new trace will begin. Pressing "E)XIT" returns the user to the engineering display. While scrolling through the 34 pages of trace data. If a spike is present on the plot, its time can be roughly measured by counting the samples. This will have to be done manually at this time but an automatic time measurement system using cursors could added later. There is a problem and that is if the data rate is very slow and the number of words per frame is very large, the time to fill the buffer will be quite long. For example, if the words were 16 bits wide, the number of words per frame were 200, and the bit rate was 1Khz, it would take 3.2 seconds per frame which equates to

53.33 minutes. For this reason, the "T)RACE" algorithm will display a page when it is ready and not wait until the memory is full. In fact, the user can see the data being plotted as it is read in if he/she keeps scrolling fast enough. A more reasonable set of data such as 10 bits per word, and 30 words per frame at 25 Khz would yield one frame per .3 sec or 5 minutes to fill memory. This is not a limitation of the PCM monitor system, only the data rates and PCM specifications.

IV. Test and Evaluation

Methodology

The PCM monitor system was tested under laboratory conditions using basic equipment such as an oscilloscope, logic probe, and volt meter etc. To test the PCM processor software, a signal generator and logic analyzer was used. The system was tested and analyzed in three steps. First, the PCM processor was tested as a system. Then the display processor software was tested using patches and canned data. The last was system test utilizing both the PCM processor and the display processor together. In this method, the success of the final test was assured with a high confidence factor.

Testing of each hardware section of the PCM processor was done during the design and construction phase of the project. This greatly reduced the risk during the system test. In fact, the only problems encountered with the PCM processor was software which was corrected. In order to verify that the PCM processor was working, a preset signal was input to the system. A Hewlett Packard HP1630D logic analyzer which incorporates a disassembler was used to analyze how the system reacted to the signal. See below for the results of the test.

The display processor software was tested incrementally as is was written. After each program in the system was completed, several short diagnostic programs were written to analyze the data in the systems data files. Once verified, the entire software system was tested using actual parameters from a recently conducted flight test.

The PCM monitor was tested as a system in the laboratory using the signal generator, logic analyzer, and the display processor itself.

Testing the System

The primary thrusts of the test were to see whether the system would synchronize with various frequencies, determine the maximum frequency it could accept, and determine the maximum number of non-varying data bits could be sent within a frame before losing synchronization.

To start the test, the data base was set up with the display processor to reflect no alarms, a minimum range of 0 and a maximum range of 65235. The purpose for this was to display raw data and not data which is normalized to some minimum or maximum. Next the PCM system frame specifications were set up for a normal PCM data stream utilizing the auto bit rate calculation, 10 data words of 16 bits each, NRZ PCM type, normal polarity, and LSB first. Two synchronization words used were $5400, and $5454. For initial test purposes, all data words were set to the value of 1 in the signal generator and the bit rate was set to around 25Khz. The results were as follows.

The signal generator was set to begin trace just after the bit rate calculation at point 1 in figure 39 to determine what the word count was. Without going into too much detail, a trace is set into the logic analyzer by programming in a certain address or data value. When the analyzer sees this value, it begins capturing all data until its buffer of 1024 data points is full. Each data point contains both address and the data bus information of the microprocessor under test. The HP1630D can then display this data in various formats but the one

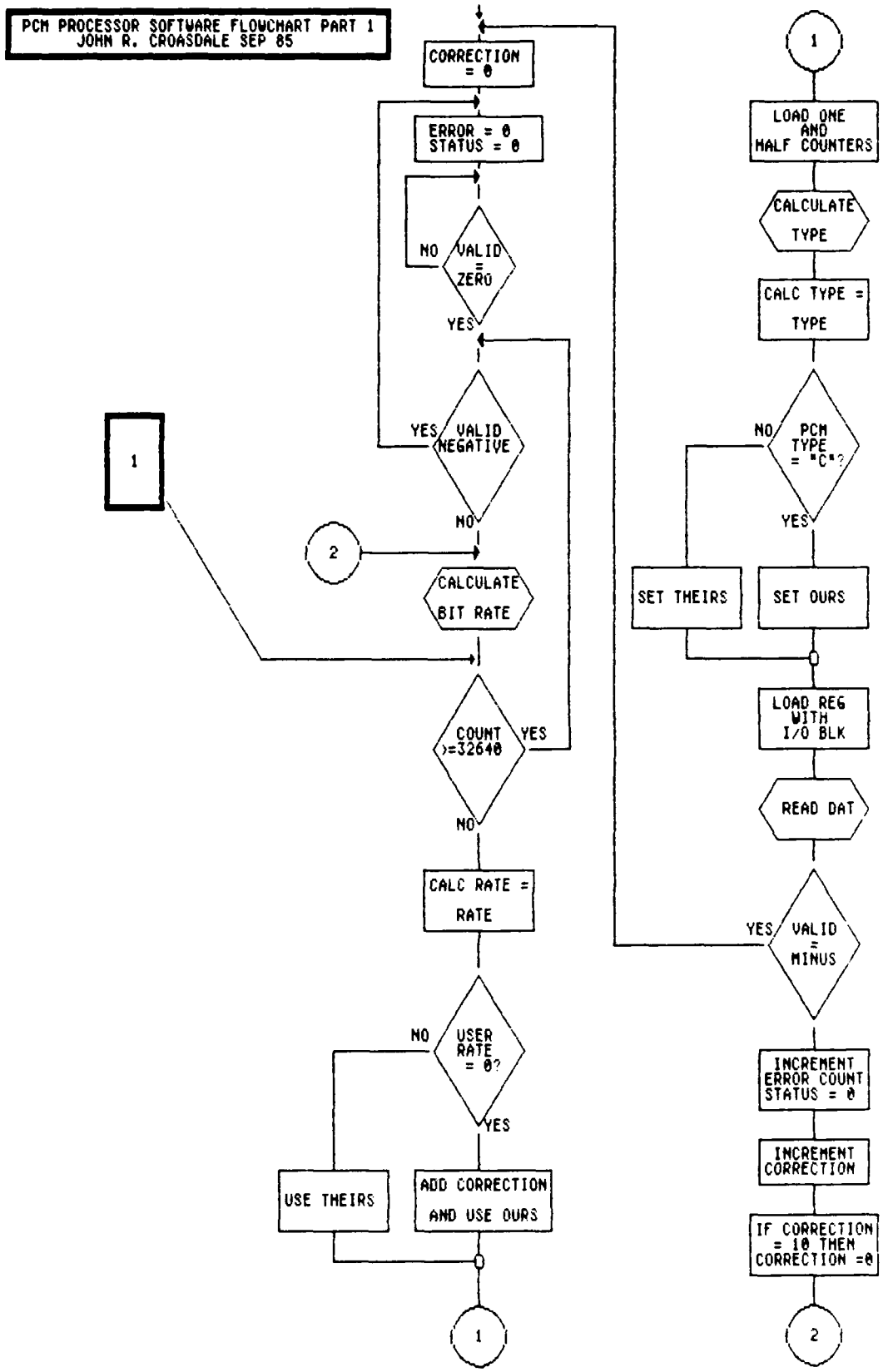


Figure 39. PCM Processor Flowchart Part 1

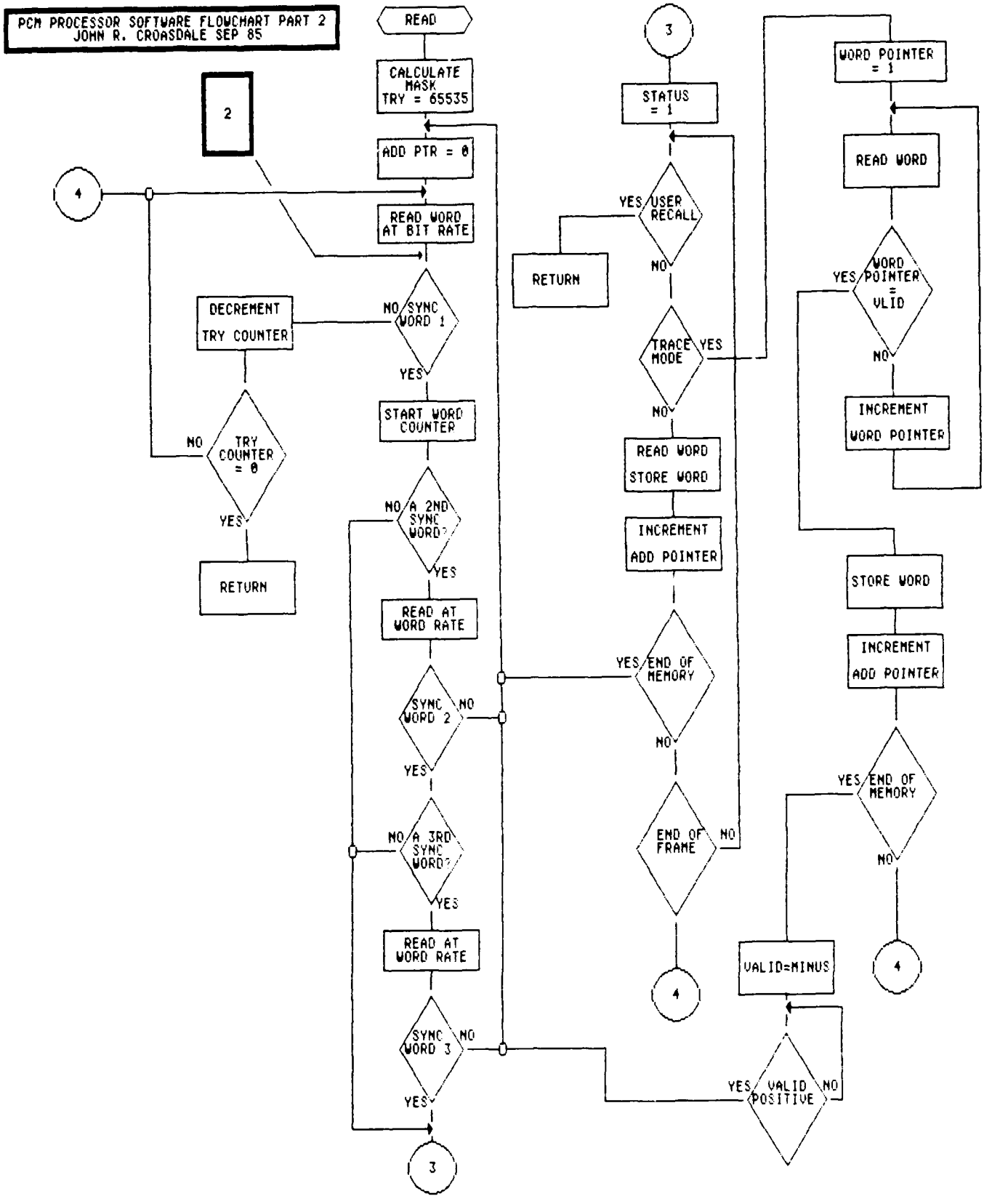


Figure 40. PCM Processor Flowchart Part 2

used in this test was the dissassembly display.

By setting the trace point at position 1 shown in the flowchart, the 65235 iterations of the bit rate calculation loop is avoided. The trace revealed that the routine returned with a value of $276 hex, 630 decimal. If the signal were exactly 25Khz, the count should have been 640 but since the signal generator used an analog setting for bit rate (a pot), the frequency could not be set exactly. It was measured on the oscilloscope to be just under 25Khz which was commensurate with the reading of 630. The trace showed that the half counter was loaded with $13B and the one counter with the $276. Looking at the status page of the display processor, 25.39khz was displayed which gave a good indication that the auto bit rate calculation was working.

The next trace was begun at point 2 in figure 40, the beginning of the critical loop where the first sync word was being searched for. The logic analyzer showed the data bits being shifted in, one bit at a time. When the $5400 (0101010000000000) started in, the first word seen on the logic analyzer was an $8000. The next was $4000 ($8000 shifted one bit to the right). Next came an expected $A000 (1010000000000000). Three shifts later the magical $5400 appeared and the trace exited the critical loop and started the word counter, just as predicted. The next word read in was a $5454, the second sync word. The routine jumped to the routine which reads in the data and places it into the buffer memory. The system was working!

The logic analyzer has a chart display which looks at the entire 1024 samples at one time in a graphical display which shows collected data all at once. In this case, is was used to show address bus

utilization. Using this display, it was clear to see when the system was in sync, and when it wasn't. For example, when the software is in a tight loop, the addresses present a regular pattern on the display. When the system was in sync, the display had a completely different appearance as the address range was now spanning the entire I/O buffer area instead of just a small section of the program.

Looking at the engineering data page of the display processor, all ten words reflected the value of 1 for all data types except switch which was "ON".

To determine the maximum frequency the system would respond to, the bit rate was increased gradually. It broke lock almost immediately but re-calculated the bit rate and re-synchronized. When the word count reached about 100, the system would not lock up again. A word count of 100 represents a bit rate of 160 Khz using the present system clock. This was apparently the upper limit of the system with the present software configuration.

The last items to test was the walk error or the number of non-varying data bits (sequential 1's or 0's) which could be sent without synchronization problems.

The walk error is the prime reason, other than loss of signal, that the PCM monitor will lose synchronization with the PCM bit stream. It is caused by the difference between the actual bit rate, and the generated one. It is only encountered during a long series of successive 1's or 0's for NRZ, and never for Bi-Phase. The term walk error is actually a misnomer in that the larger it is, the more non-varying bits the system will tolerate. The following discussion

concerns NRZ signals only. See figures 41 and 42.

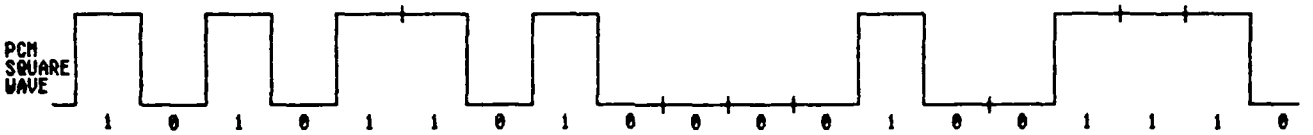
Figure 41 shows a typical section of an NRZ-L PCM waveform. The top section shows the relationship between the sync pulses generated within the shift generator, and the shift pulses generated by the shift generator. The PCM signal is synchronized by the sync pulses which occur at each positive transition of the incoming signal whether it be NRZ or Bi-Phase. During a long series of 1's or Ø's, the signal is not re-synchronized and any error between the actual bit rate and the calculated bit rate is accumulated. The maximum error can theoretically be adjusted to one resolution of the 16Mhz counting clock or 62.5ns by careful analysis of the output of the BIT TIMER, an iterative process, or accounting for the delays in the counting circuits. In practicallity, however, an accuracy of 2 to 3 resolution elements is all that is expected depending on the stray delay times, and jitter of the incoming PCM stream. The PCM processor software has been programmed to do an iterative process and return to the user a correction factor that reflects the quality of the waveform.

The lower section of figure 41 shows a consecutive series of 1's and the resultant walk error. The figure shows a gross example where the number of counts per bit time is only 7 to allow examination. Figure 42 shows the derivation of the equation to determine the number of consecutive 1's or Ø's needed to generate a bit error because of the walk error.

To measure the walk error, the frame setup was changed to reflect 20 data words at 16 bits each. This would yield a maximum of 20 X 16

SHIFT PULSE AND WALK ERROR ANALYSIS	PART 1 SEP 85

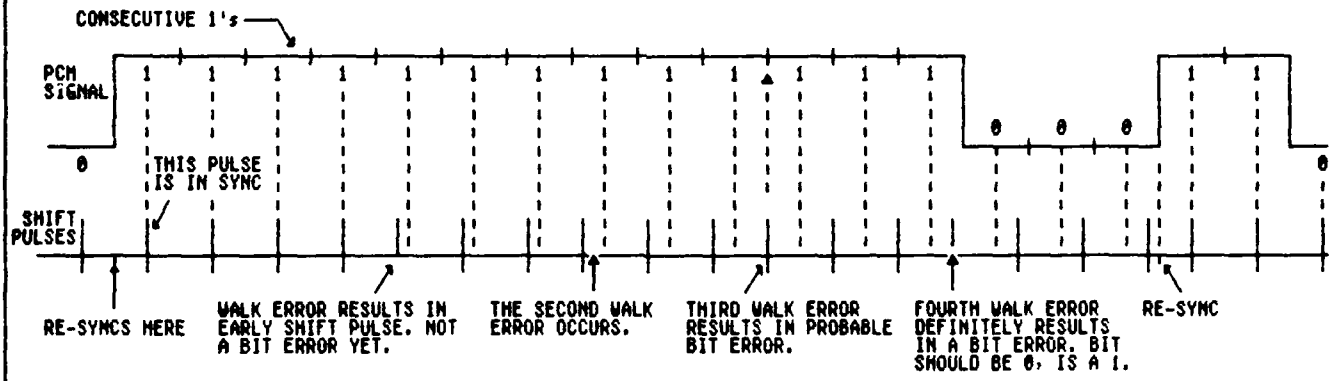
EXPLANATION OF SYNC AND SHIFT PULSES



SYNC PULSES OCCUR AT EACH POSITIVE TRANSITION OF THE PCM INPUT SIGNAL. THIS GENERATES A SHIFT AT .5 TIMES THE BIT RATE. AFTER THAT, AND UNTIL ANOTHER SYNC PULSE OCCURS, SHIFT PULSES OCCUR AT THE BIT RATE - THE WALK ERROR.

THE WALKING ERROR:

A CONSECUTIVE SERIES OF 1'S OR 0'S WILL EVENTUALLY CAUSE THE BIT STREAM TO LOSE SYNC BECAUSE OF THE WALK ERROR. BECAUSE OF THE DIFFERENCE BETWEEN THE DIGITAL COUNT, AND THE REAL, INFINITE RESOLUTION COUNT. WALK ERROR IS MAXIMIZED AT ONE CLOCK PER PCM BIT TIME. SEE BELOW. REAL COUNTS ARE SHOWN AS DOTTED LINES. DIGITAL (ACTUAL) COUNTS ARE SOLID.



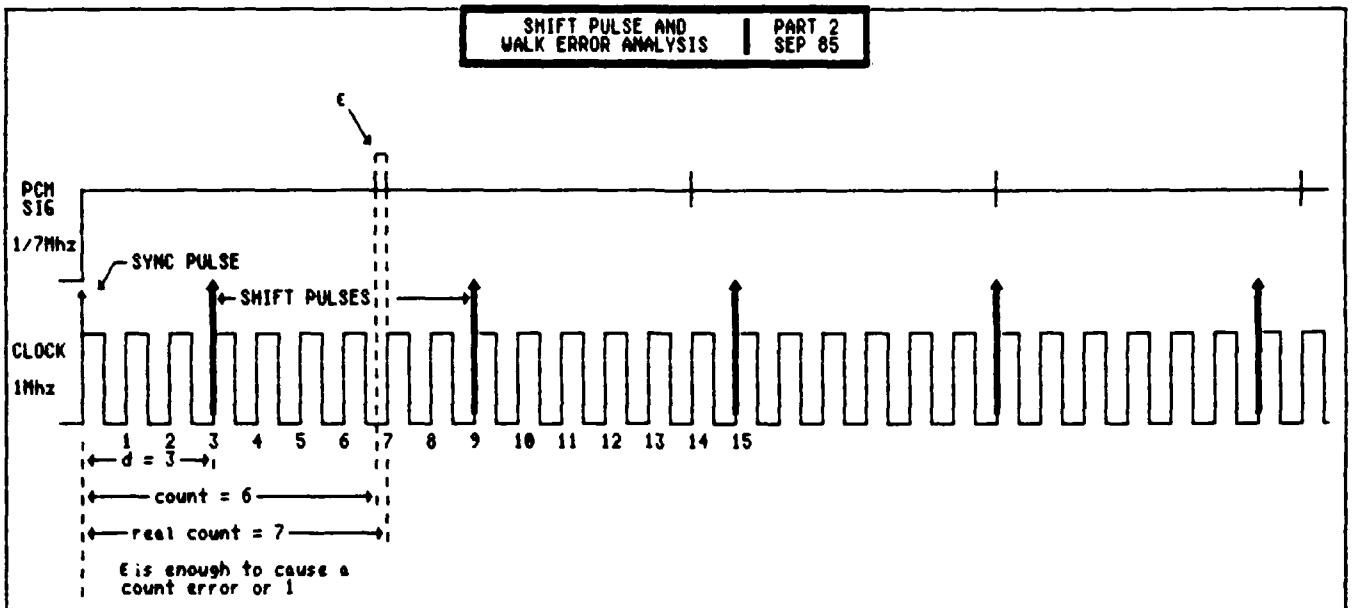
TO CALCULATE N, THE NUMBER OF CONSECUTIVE 0's OR 1's DECODED BEFORE A BIT ERROR IS FOUND IS

$$N = \mathrm{INT}\left[\frac{\left[\frac{\text{CLK FREQ}}{\text{PCM FREQ}} - 1\right]}{2}\right]$$

(see part 2 for derivation of N)

FOR A CLK OF 16Mhz AND A PCM FREQ OF 100Khz, $N = \mathrm{INT}\left[\frac{\left[\frac{16000000}{100000} - 1\right]}{2}\right] = 79$

Figure 41. Walk Error Analysis Part 1



THE GOAL IS TO DETERMINE "N", THE NUMBER OF CONSECUTIVE 1'S OR 0'S DECODED BEFORE RECEIVING A BIT ERROR. BY INTUITIVE REASONING, WE CAN DETERMINE THAT THE WALK ERROR IS THE DIFFERENCE BETWEEN THE REAL COUNT AND THE ACTUAL COUNT. THIS ERROR OCCURS BECAUSE THE REAL COUNT IS A REAL NUMBER AND THE COUNT IS AN INTEGER. IN FACT, THE COUNT IS THE INTEGER PORTION OF THE REAL COUNT = INT(REALCOUNT). IF THE PCM FREQUENCY WAS AN EXACT MULTIPLE OF THE CLOCK FREQ, THEN COUNT WOULD = REALCOUNT AND THERE WOULD BE NO WALK ERROR. TO DETERMINE THE MAXIMUM WALK ERROR, CONSIDER THE CASE WHEN COUNT = REALCOUNT - ε WHERE ε IS CLOSE TO BUT UNEQUAL TO ZERO. THEN COUNT WOULD BE INT(REALCOUNT) OR REALCOUNT + ε - 1.

IN THE ABOVE EXAMPLE, THE CLOCK AND PCM FREQUENCIES ARE CLOSE TO ALLOW FOR AN EXAMINATION OF WHAT IS GOING ON. IN THE REAL WORLD, THE CLOCK TO PCM RATIO WOULD BE MUCH GREATER.

WALKERROR = REALCOUNT - COUNT

REALCOUNT = CLOCK/PCM

COUNT = INT(REALCOUNT - WALKERROR) FOR WORST CASE, ALSO USE CLOCK AND PCM FREQUENCIES

DEFINE R = RATIO OF WHERE THE FIRST SHIFT PULSE IS TO OCCUR TO THE BIT TIME

d = INT(R * COUNT)

SINCE d IS THE NUMBER OF CLOCK CYCLES TO WHICH THE WALKERROR MUST BE APPLIED TO (the starting point etc.)

N = d/WALKERROR

IF WALKERROR WERE 0, N WOULD BE UNDEFINED OR ESSENTIALLY INFINITE

THE MAX LIMIT OF THE WALKERROR IS 1 SO THE MAX N WOULD BE

N = d

N = INT(R * (INT(CLK/PCM) - 1))

TO GET THE BEST POSSIBLE SECTION OF THE PCM WAVEFORM, WE CHOOSE TO LOOK A THE PCM SIGNAL AT THE CENTER OF THE BIT TIME SO R = 1/2 AND

N = INT(.5 * (INT(CLK/PCM) - 1))

FOR THE SAMPLE ABOVE, CLK/PCM = 7 SO N = INT(.5 * 6) = 3

FOR A PCM SIGNAL OF 100hz AND A CLOCK OF 16Mhz, N = INT(.5 * 160-1) = 79

Figure 42. Walk Error Analysis Part 2

or 320 non-varying bits. Each data word was programmed to zero in the signal generator.

The frequency was reduced again to 25Khz and the previous test was repeated. The word count was now 635 and the system synchronized. At this rate, the walk error should be around 319 according to the calculation. This error, of course, is the worst case so there shouldn't be any problem with synchronization.

When the rate was increased to beyond 40Khz, the system became very touchy to frequency and began to lose sync. This was expected because in the walk error analysis, if the bit frequency were an exact multiple of the 16Mhz clock, there would be no walk error. Any variation from the optimum would increase the risk of problems.

To determine the walk error at 100Khz, the signal generator was set to 100Khz which caused the PCM monitor to break lock. As each data word was modified to a non-zero value starting in the middle and working towards the ends the synchronization of the system was checked. If the signal synchronized, the last data word changed back to zero and a new one was which gave a larger number of non-varying sequential bits was modified. Finally the limiting bit was found and the walk error was counted. In this case it was 124. The calculated maximum walk error was 79. Remember that the walk error is a misnomer. The larger it actually is, the less the error and the more non-varying bits the system will accept before losing sync. The value of 124 was encouraging. Varying the frequency a little, however, caused the system to break lock again.

By setting the data word values so that only 79 sequential bits

were zero, there was no problem with synchronization even when the frequency was shifted from 90 to 110 Khz. During one test, the system actually remained in synchronization after 320 sequential zeroes at a rate of 132Khz. Moving the frequency adjust control just a little, however, caused the system to break lock and not recapture.

There The conclusion is that the walk error calculations are correct and the system can reliably accept 79 sequential non-varying bits at data rates up to 100Khz without breaking lock.

There was one problem observed from time to time during the test and that was that at certain undocumented frequencies, the system would not synchronize. This occurred even when the walk error should have not been a factor. By varying the frequency a minor amount the problem disappeared. This problem occurred in all frequency ranges and indicates that there may be a race problem of some sort in the word processor. Inspecting the dissassembly of the PCM processor software indicated that when this occurred, the value read from the word processor was all F's as if the registers were not enabled. Other than noting the difficulty, the problem was not persued and will be left to follow on investigations.

Final testing was done by setting random data words at varying bit rates using different POLARITY and ORDER. The system worked in each case and the testing was terminated.

V. Recommendations and Conclusions

Although the development and test of the PCM monitor system was a success, there is need for improvement in both the PCM processor and the display processor.

The PCM processor could be improved in at least four ways listed below.

1. Increase the microprocessor clock frequency to 12 Mhz or more to allow a shorter execution time of the critical loop and the subsequent increase in bit rate capability.
2. Increase the bit count frequency to at least 32 Mhz to minimize the walk error.
3. Include a self test feature.
4. Provide an improved bit rate calculation utilizing a statistical method rather than an iterative one.
5. Continue development to eliminate the frequency sensitivity problem.

The state-of-the art clock frequency for the MC68000 microprocessor is presently 12Mhz and is predicted to be 16Mhz before long. The PCM processor should be built on a printed circuit board to take advantage of this improved capability.

Since the bit timer counting clock is independent from the processor clock, the maximum frequency should be used to reduce the walk error. The only limiting factor is the speed of the IC devices. Utilizing new high speed devices like the Emitter Coupled Logic (ECL) series and making the PCM processor into a printed circuit board, there is no reason that the bit counting clock frequency could not be increased to above 50 Mhz. The software would have to be modified, however, to account for the larger word count figures.

A self test feature could be added to present canned displays to the test director convincing him that the PCM monitor is functioning

correctly when problems are indicated with the PCM stream. Such a self test could be an on-board simulated PCM signal generated using shift registers and counters.

The current method to zero in on the actual word count that provides a reliable synchronization is iterative. This causes the system to take longer to provide reliable data than is actually necessary. A statistical method utilizing data collected during the bit rate calculations could provide a better best guess starting point than the current method does using the minimum word count.

The display processor was not meant to be the final solution. As the system matures, many ideas will occur which will need to be included to provide a better window into the test system. The display software is currently maxed out in both size and execution time. The EXECUTE program, for example, leaves only approximately 2000 bytes of memory for improvements out of a total of 22800 when the computer is empty. In addition, the software is so slow that each display takes about 1.2 seconds to update. That relates to an update rate of 0.83Hz.

To improve both size of the program and execution speed, an optimizing compiler should be used to compile the EXECUTE program. This should reduce its size by removing the many included comments. In addition, a compiled program runs from 10 to 1000 times faster than a BASIC program depending on the operation involved. The average increase in speed judged from other compilers is around 200% so the update rate should be around 16.5hz as compared to 0.83Hz.

Such improvements would allow more calculations on the data before display and better display pages showing more data at once such as

multiple graphs etc.

Conclusions

The development of the PCM monitor was a success. It met or exceeded all the requirements specified at the outset of the project. It has been demonstrated in a laboratory environment to lock on to a PCM data stream of unknown frequency and display the data in various formats at bit rates exceeding 100Khz. Its automatic feature of PCM type determination was a success as was its ability to re-synchronize when the bit rate was changed from one frequency to another.

The system will provide a valuable test monitoring capability when modified for flight. It now provides the basic tools which will inspire increased development and capability.

This low cost system could be very easily adapted to the mission of the 4950th Test Wing with minimum of effort and provide the test director with the real time monitoring capability needed.

Bibliography

1. Owen, Frank F. E. PCM and Digital Transmission Systems. New York, McGraw Hill, 1982

2. High Speed CMOS Products. Integrated Device Technology, Inc, 1985.

3. TTL Data Book. Mountain View, CA. Fairchild Camera and Instrument Corporation, 1978.

4. Kaine, Gerry, Doug Hawkins, and Lance Leventhal, 68000 Assembly Language Programming; Berkeley, CA, OSBORNE/McGraw Hill, 1981.

5. Leventhal, Lance A. 6809 Assembly Language Programming; Berkeley, CA, OSBORNE/McGraw Hill, 1981.

6. Motorola, 16 Bit Microprocessor User's Manual, Englewood Cliffs, N.J., Prentice-Hall, 1979.

7. Getting Started With Extended color Basic, Tandy corporation, FortWorth, Texas, 1984.

8. Radio Shack TRS80 Color Computer Disk System Owner's Manual and Programming Guide, Tandy corporation, FortWorth, Texas, 1981.

Appendix A

REAL-TIME FLIGHT TEST PCM DATA ACQUISITION MONITOR

PROGRAMMER'S MANUAL

BY

JOHN R. CROASDALE, LTC, USAF

Programmer's Manual

Overview

This manual is intended to provide the programmer with enough information about the software of the PCM processor and display processor to allow modification at a later date. Each program is described and contains flow chart with an explanation of how the program works. The program's overall command structure is shown in figure 43 and the listings are shown in appendix C of the thesis (1).

Some commonly used terms and definitions are included here for reference.

1. Data Word: The actual word as it appears in the PCM data stream. Data word #1 is the first word after the synchronization words etc.

2. Parameter: The test item which is to be measured. Data word #1 could be the 3rd parameter in the test etc.

3. Wraparound: The feature which folds the end of a list to the beginning and reverse. That which is incremented beyond the bottom of a structure reverts back to the top.

4. Case: A structure which decodes an input of some store and branches to a particular routine depending on the code.

5. Quit: A function used to terminate a program and return to the MAIN program.

6. Exit: A function used within a program to exit from a certain section back to the calling routine.

Flow chart standards used in this manual are depicted in figure 44. Please refer to the main thesis (1) for an overall description of the system.

The programs described in this manual are:

1. PCM MONITOR ver 1.0 - Assembly program which does the function

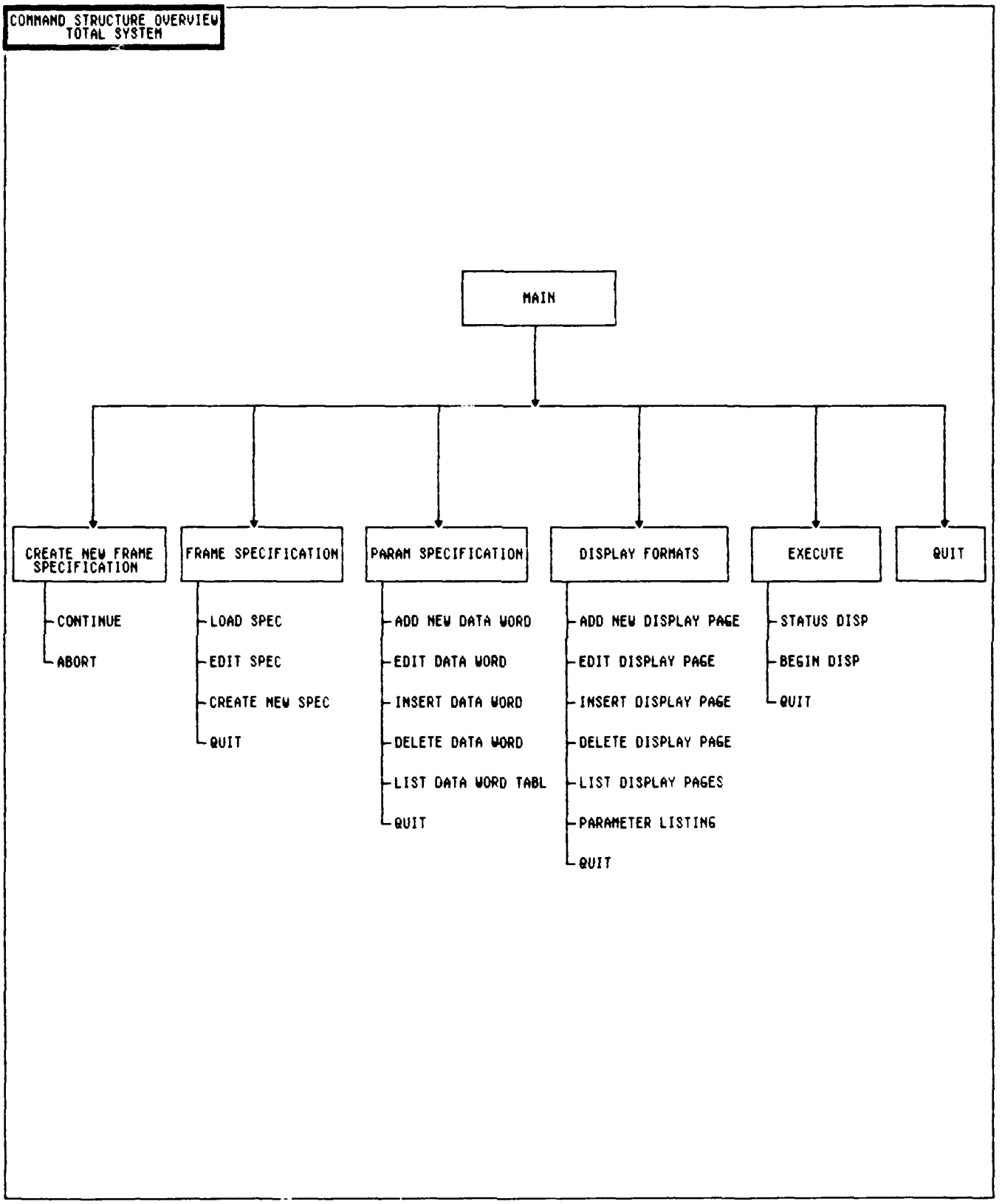


Figure 43. Command Structure Overview

for the PCM processor.

2. MAIN.BAS ver 1.Ø - BASIC program which acts as main coordinating program for the display processor.

3. FSETUP.BAS ver 1.Ø - BASIC program which sets up frame specifications.

4. PSETUP.BAS ver 1.Ø - BASIC program which sets up the parameter data base.

5. DSETUP.BAS ver 1.Ø - BASIC program which sets up the displays.

6. EXECUTE.BAS ver 1.Ø - BASIC program which displays data to the user.

7. CONVERT.BIN ver 1.Ø - Assembly program which does type data type conversions.

PCM Processor

The flow chart for the PCM processor software is shown in figures 45 and 46. All information transfer between the PCM processor and display processor is through the I/O control block shown in table 3.

The most important location in the I/O control block is the valid flag. With this flag, the user tells the PCM processor to start operations, quit operations, move to the trace mode or return to the continuous mode. Upon startup, the PCM processor sets the valid flag to all F's, (all bits high). When the display processor inputs all the PCM stream data into the I/O control block, it signals the PCM processor to start looking for the PCM signal by setting the valid flag to Ø. If it wants it to stop, the valid flag must be set to some negative number (bit 15 must be set). If the trace mode is desired, the display processor must set the valid flag to some positive value. That value must be the number of the data word in the PCM stream to do the trace on. No error checking is accomplished and if a data word

I/O control block

$C7FE	VALID FLAG	INPUT: Ø=CONTINUOUS MODE 1-ØØFF=TRACE MODE, NUMBER=DATA word > $7FFF=STOP AND WAIT
$C7FC	USER BIT RATE	INPUT: Ø=AUTO, <>Ø=RATE IN word COUNT
$C7FA	PCM TYPE	INPUT: "C"=CALC, "N"=NRZ, "B"=BI-PHASE
$C7F8	word LENGTH	INPUT: HEX VALUE 8 - 16 BITS
$C7F6	FRAME LENGTH	INPUT: HEX VALUE Ø - $FFFF
$C7F4	FRAME SYNC1	INPUT: HEX VALUE
$C7F2	FRAME SYNC2	INPUT: HEX VALUE
$C7FØ	FRAME SYNC3	INPUT: HEX VALUE
$C7EE	ERROR COUNT	OUTPUT: NUMBER OF SYNC ERRORS
$C7EC	POLARITY	INPUT: "N"=NORMAL, "I"=INVERSE
$C7EA	FRAMES/BUFFER	INPUT: HEX VALUE
$C7E8	CALCULATED TYPE	OUTPUT: "N"=NRZ, "B"=BI-PHASE
$C7E6	CALCULATED RATE	OUTPUT: RATE IN word COUNT
$C7E4	CORRECTION	OUTPUT: NUMBER ADDED TO COUNT TO SYNC IN
$C7E2	STATUS	OUTPUT: Ø=NOT IN SYNC, <>Ø=IN SYNC
$C7EØ	DATA ADDRESS	OUTPUT: ADDRESS OF LAST DATA VALUE WRITTEN

The addresses used in this table are with respect to the PCM processor. They are essentially the same for the display processor except they are mapped $2000 addresses lower. Addresses for the I/O control block according to the display processor range from $E7E2 to $E7FE.

Table 3. I/O Control Block

number larger than number of words per frame is entered, the system will lose sync and reset.

The user bit rate location is used to pass PCM rate to the PCM processor. If the user wishes to use an automatic calculation of the bit rate, he/she places a Ø in this location. If a manual rate is desired, the rate in word count must be entered. The display processor converts the user input frequency to word count and places it here if required.

The word length contains the number of bits per words. The PCM processor uses this value to create a mask to delete the unwanted bits from a data word value. Actually any word length value will be processed but for word lengths of less than 8, the time to process between reading words may not be sufficient and the system will lose sync. Values above 16 will be yield the same results as 16. This is because the mask generation routine continually shifts 1's in the mask register from the left. More than 16 shifts will continue to keep the register full or 1's regardless. See the program listing.

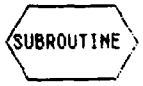
The frame length is the number of words per frame not including the synchronization words. The PCM processor uses this value to set counters which will alert the system when to look for the next sync word/s. If the wrong value is entered, synchronization will occur but for only one frame at a time as the sync words will be in a different place with respect to where the system thinks they are.

The next three address should contain the sync word values. The first location should contain the first value, the second is next and the last address should contain the third. A special note here to the

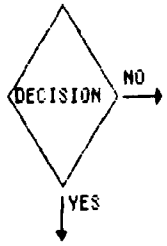
FLOWCHART SYMBOLS AND EXPLANATIONS
JOHN R. CROASDALE SEP 85

THE FOLLOWING FLOWCHARTING STANDARDS AND SYMBOLS USED THROUGHOUT THIS SECTION.

DEFINES A PROCESS IN GENERAL TERMS. LINE NUMBERS ARE GIVEN WHEN APPLICABLE. SOMETIMES A SMALL SUBROUTINE ARE INCLUDED. LARGER SUBROUTINES HAVE SEPARATE FLOWCHARTS.



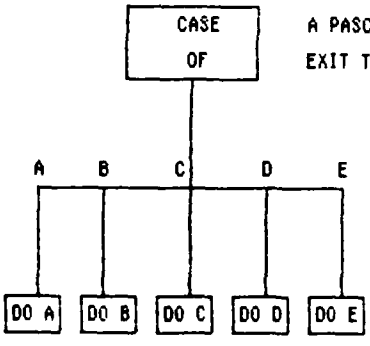
A SEPARATE ROUTINE WHICH DOES A FUNCTION. OTHER SUBROUTINES MAY BE CALLED FROM WITHIN. ALL SUBROUTINES ARE EXITED WITH A RETURN PROCESS BLOCK.



A DECISION BLOCK WHICH HAS A BRIEF STATEMENT OF THE DECIDING CRITERIA. YES AND NO MAY MAY EXIT FROM ANY OF THE FOUR CORNERS OF THE BLOCK.

AN ON PAGE CONTINUATION OR GOTO INDICATION. SOMETIMES IT'S IMPOSSIBLE TO SHOW ACTUAL FLOW LINES AND STILL KEEP THE FLOWCHART UN-CLUTTERED. UNIQUE NUMBER SETS OF TWO MUST BE ADHERED TO.

OFF PAGE EXIT OR ENTRY CONTINUATION BLOCKS WHICH PASS CONTROL TO ANOTHER PAGE OF THE SAME SET. THE NUMBERS MUST BE UNIQUE TO THE SET BUT NOT NECESSARILY TO THE SYSTEM. NUMBER 1, FOR EXAMPLE ON PART 1 OF A FLOWCHART MUST BE CONTINUED TO ANOTHER PART OF THE SAME FLOWCHART.



A PASCAL-LIKE CASE STATEMENT WHERE THE DETERMINING FACTOR, (KEY etc.) DECIDES WHICH PATH TO EXIT THE STATEMENT FROM.

Figure 44. Flowchart Symbols and Explanations

the programmer who wishes to modify the display processor software: The polarity must be applied to the sync word definitions before sent to the I/O control block. The PCM processor reads in all data in normal format and inverts it, if necessary, if inverse polarity is in effect. There is enough time to do this when reading data at the word rate but not at the bit rate. If the polarity is inverse and the user inputs a sync word value of "00001111", it should be converted to "11110000" before sending it to the I/O control block.

Location $C7EE is an output location in which the PCM processor sends the error count to the display processor to be used in the status page in program EXECUTE. The error count is incremented each time synchronization has been established and then lost. If no signal were present, no errors would be reported. In other words, an error implies that the system was in sync at some time. If the system wasn't in sync, there could be no error.

Polarity is input to the PCM processor through location $C7EC in the I/O control block. Polarity is simply the reversal of the voltages and the bit values they represent. Normally all PCM signals represent a "0" as a low voltage level and a "1" as a high level. In bi-phase, a transition from low to high normally represents a "0" and a transition from high to low represents a "1". In some systems tested in the past on flight test programs, the bi-phase transition meanings were reversed. The polarity capability was added to allow for such non-standard systems.

The frames/buffer input was added to avert additional division

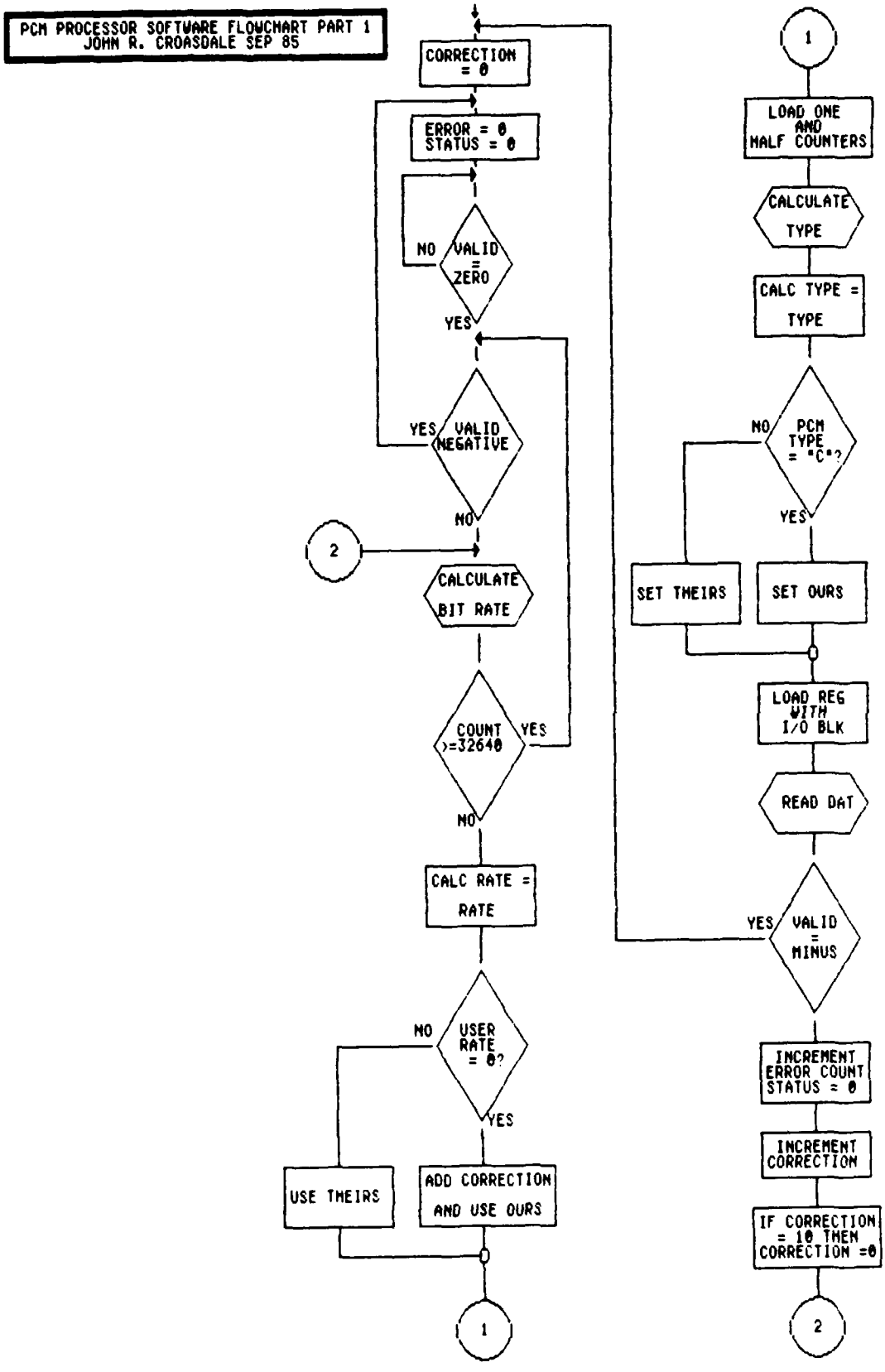


Figure 45. PCM Processor Flowchart Part 1

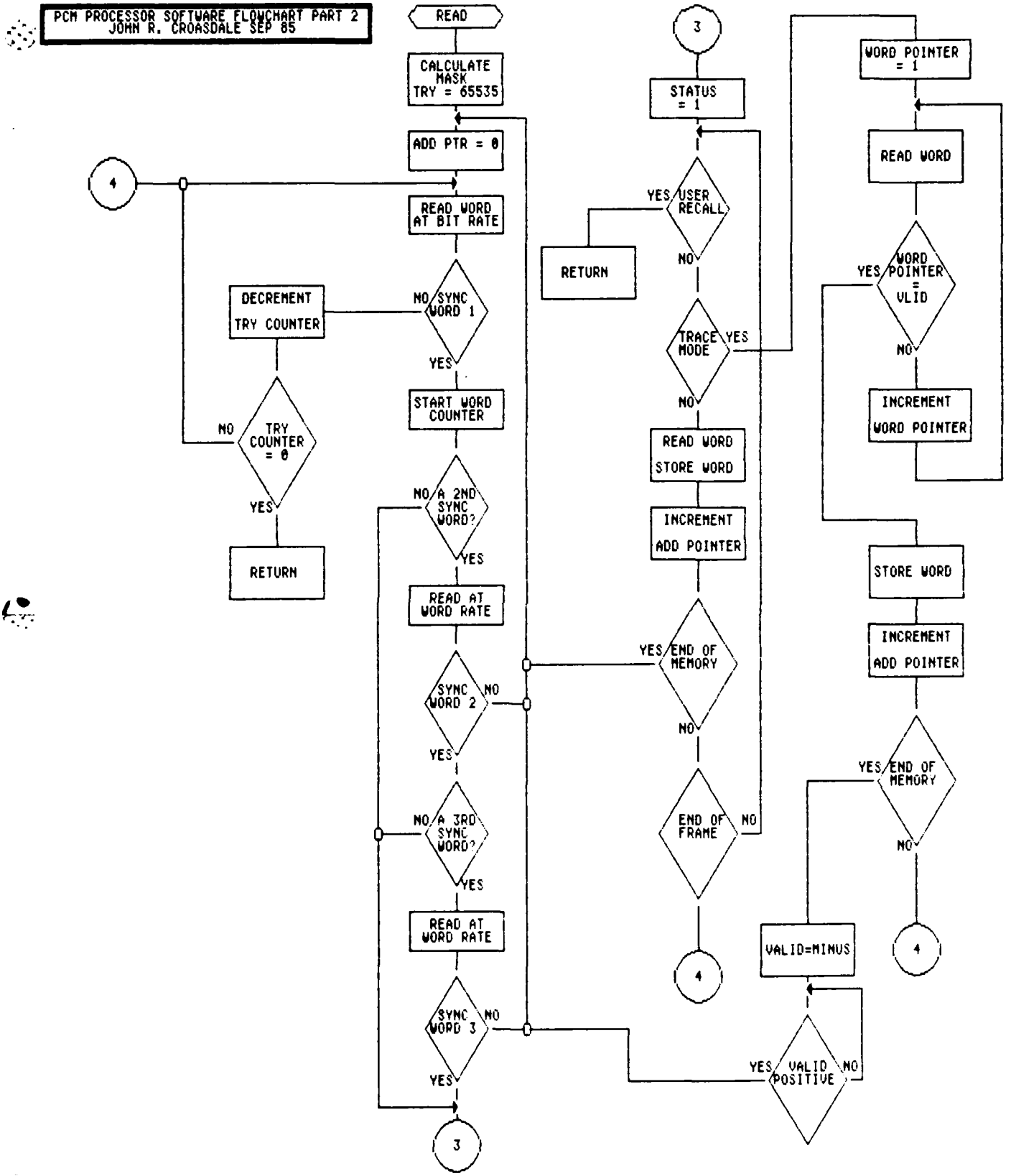


Figure 45. PCM Processor Flowchart Part 2

routines in the PCM processor software. The calculation is done in the display processor software to take advantage of the powerful instructions of the BASIC language. The value is used to determine when the buffer is full and to start over.

The PCM processor outputs the calculated PCM type in the next location, $C7E8. Whether or not the user wants to use a preset PCM type, the PCM processor does the calculation anyway and returns it to the user. The value is displayed on the STATUS page of the display processor.

Likewise the calculated PCM rate is returned in location $C7E6. The PCM rate is calculated regardless of whether the user wishes it to be or not, and returned on the STATUS page. This value can be used to determine if the PCM stream is at the proper bit rate or not.

The correction value is returned in location $C7E4 and is indicative of the jitter in the PCM stream. The CALCULATE RATE routine finds the minimum number of counts between two positive going pulses. For an entirely stable PCM data rate, the minimum will always be the same within one count. If jitter is present, the minimum count will be somewhat less than the actual bit rate times two. Note that the count between two positive going signal pulses is twice that which occurs during one bit time. If the bit rate is high enough and the jitter is bad enough, the walk error explained in section IV of the thesis will become substantial and cause the system to break sync. The PCM processor corrects for this by adding 1 to correction each time synchronization is lost. If the value becomes larger than 10, it is reset to 0. In this way, the system keeps on track with even with a

widely varying PCM signal. The maximum value of 10 was chosen as an educated guess and tested in the laboratory. The value of the correction counter is returned in the correction location of the I/O control block

The status location of the I/O control block simply returns a value of non-zero (true) to the display processor) when the PCM signal is in sync, and zero (false) when not. When the display processor finds a false value in status, updates of data values are halted.

The data address is the address pointer to the last I/O buffer location written to. It is used in the TRACE routine to synchronize the display processor with the PCM processor. As data is read in, the pointer is updated to tell the display processor where in the trace the system is.

The remaining locations in the I/O BUFFER are for data word values. As the system controller reads in the data words, it stores them here beginning at address $C000 and ends at the integer value of $C000 + word length times frame length. The maximum value assures that the buffer will not be overfilled.

The system controller interfaces to the rest of the PCM processor system through the remaining addresses listed in table 2. These were described fully in section I of the thesis and will not be repeated here. See appendices B and C for flowcharts and listings of the PCM processor software.

The continuous mode of operation provides the user with a continuous update of input PCM data regardless of whether or not the display processor can read it all. If the display processor is slow,

data will be lost as the PCM processor will keep writing data to the buffer whether it is read out or not. In other words, it is up the the display processor to use the data as it sees fit.

Trace mode, on the other hand, is synchronized to the display processor in that it fills the memory buffer with sequential frames of only one data word, keeping the data address pointer in the I/O control block updated at all times. When the buffer is full. the PCM processor sets the valid flag to minus and waits for a user input telling it to begin another trace. This is done by by resetting the valid flag with another data word value. If the user sets the valid flag to Ø, system is returned to the continuous mode. Note that the user cannot directly cause the system to reset from this position. To do this, a Ø must first be placed in the valid flag to exit from the wait loop, then a reset command must be given.

Part 1 of the PCM processor flowchart shows the general control flow for the system. The first block initialized the correction factor to zero. This is only done during initial startup, and when the user forces a restart after the system has been running. The reason for this is that the correction factor is determined over a long period of time and should not be reset indiscriminately. The user may want to set up the system for an alternate signal, in which case, a new set of PCM data will be entered. The user must then force a restart to bring up the system.

The next few blocks check for a valid signal from the user saying that the I/O control block has been set. The user starts the system by placing a positive value in the valid flag. If the value is Ø, the

system runs in continuous mode. If it is non-zero, it uses this value in trace mode to identify the data word to trace on. Part 2 of the flowchart shows this feature.

After the go ahead, the system does a bit rate calculation. If the count is larger than $7F00, 32640, the bit timer has maxed out. Note in the listing that the word count is divided by two in the CALCULATE BIT RATE subroutine before returning. The bit timer actually maxes out at $FF00.

If there is a signal, the system sends the word count value to the user through the I/O control block and checks to see whether or not to use it. The selected count is loaded into the counters and the PCM type is calculated.

The CALCULATE TYPE routine simply does a maximum word count calculation and checks it with the minimum calculated in the CALCULATE BIT RATE routine. If the maximum is two times or less than the minimum, the signal must be bi-phase and the routine returns with a "B" in the D1 register. If it is not, an "N" is returned for NRZ. This value is fed back to the user. If the user wishes to use the calculated type, a "C" will appear in the PCM type location. The program checks this and sets whatever the user desires.

The value to be placed into the word buffer address must be even or odd to set the PCM type. See table 2. The value returned is either an ASCII "N" or "B". Therefore the value is complemented and shifted to make the "N" and even number, and the "B" odd. This number is written to the word processor address which sets the PCM type.

The READ subroutine is called next and explained later. The

subroutine has two return conditions. If the valid flag is positive, then the return must be an error. If it is negative, it is a user recall. A user recall resets the system as shown but the error return increments the error counter and the correction factor. It also sets the status flag to zero showing that the system is not in sync.

It is assumed that the calculated rate is in error so the iterative process adjusts the correction factor by one. If the correction gets larger than 9, it is reset to Ø as it must be assumed that another type of error is occurring. The user can keep track of the errors by looking at the status page from program EXECUTE.

The READ subroutine looks complicated but in fact is very simple. It first calculates a mask according to the word length in the I/O control block. This mask is necessary to avoid confusion when the data words, and sync, are read from the word processor. The mask is used in the sync determination logic blocks, as well as in the read logic blocks. Consult the listing.

To find the first sync word, the PCM stream must be read, bit by bit. To keep from reading forever, a TRY counter is set to a maximum value of $FFFF. If the system reads this many bits without a sync, there must be an error of some sort and returns. The TRY counter is not cleared until READ is called from the main program. Again this assures that the system won't keep trying forever to sync on a signal without telling the user that an error has occurred. If the first sync is found but the second isn't, the TRY counter keeps counting.

To account for systems with only 1 or 2 sync words, the sync word location in the I/O control block is checked to see if there is

actually is one. A zero indicates that there is no sync word so the system bypasses the remaining synchronization steps.

When the system is in sync, the program stores a 1 in the STATUS flag to alert the user that the system is lock to the PCM signal.

As mentioned earlier, if the valid flag is Ø, the user wants continuous mode. The flowchart shows this check in the second column in figure 47.

continuous mode does a continual transferring of data from the word processor to the I/O buffer (2nd column). The code first checks to see if there is a user recall. If not, it reads a word, stores it, updates the memory address counters, and continues. At the end of each frame, the sync process is invoked and the process is started over. If at the end of memory, the address pointer is reset to Ø and the process is started over again. This continues until the system is recalled by the user, or an error occurs.

In trace mode, only one data word out of the frame is stored in sequential memory locations. Since the data word is stored in the valid flag, it is compared with a word counter started at the beginning of each frame. If there is a match, the word is stored and the address pointer is incremented and the routine returns to the synchronization portion to look for the next frame sync. At the end of memory, the valid flag is set to a negative value and waits. If the user chooses to do another trace, the value of the data word to be traced must be placed in the valid flag as before. If continuous mode is desired, a zero must be placed in the flag. If the user wants to restart the system, a zero must be inserted first followed by a negative value.

Display Processor

The software written for the display processor is in BASIC with the exception for a few routines which do data conversion for display on the engineering display page. See the Radio Shack BASIC book (2). The main thesis (1) describes the page formatting and descriptions in detail. This manual describes some of the algorithms and flowcharts to will help the programmer maintain the display processor software. Some of the easy to follow subroutines will not be discussed but are commented heavily in the software listings (1 appendix C). For convenience, all of the flowcharts for this discussion appear at the end of this manual.

Since BASIC is not a structured language, every attempt was made to overcome this deficiency. Case like structures are used throughout as the flowcharts depict. Variable assignments are of two types, those which a subroutine can use without worrying about, and those which can't be. The common value variables such as I, J, K, N etc. (mostly single letter variables) are considered expendable. They are used as counters, etc., and can be used for the most part within a routine without worrying about entry or exit conditions. The important variable, A$, is used to pass string information from one subroutine to another. Values stored in A$ change constantly so the programmer must save any data in A$ in another variable if it will be needed later.

Most of the two letter variables, arrays, and strings other than A$ are meant for one purpose, to store a piece of data throughout the program. For example, N$ always contains the name of the parameter as

it is read off the disk. N$(I) contains the name of the parameter for data word I. This is used constantly when a certain parameter is assigned to line I for display. The listing has a full explanation of these variables and how they are used. Figure 47 shows the data flow diagram for the flow of information between the software modules. This was explained in the SOFTWARE section in chapter III. The actual byte assignments in each file is shown in tables 4 and 5 below.

There are two data files in which system information is stored. These files are called PARAM.DAT, and DISPLAY.DAT. Each of these files is a direct access file which allows the programmer to access any record directly without a sequential search. It also means that each record within a file must be the same length. For PARAM.DAT, the length is 60 bytes and for DISPLAY.DAT, it is 250.

Each record in the file is stored as a single ASCII text string. Two disk access routines appear in each program except MAIN to allow data to be written or read from either file. Their names are DISK OUTPUT, and DISK INPUT. They are very simple and not shown in a flowchart but the operation is as follows. The calling routine stores a formatted string into a variable A$. This string contains the records data variables appended to each other at specific positions within the string. For PARAM.DAT, A$ = N$ + PN$ + U$ + MN$ + MX$ + AL$ + AH$. See Table 4. DISPLAY.DAT is structured the same way using the variables depicted in table 5.

After A$ has been formed, the size of the file, the name of the file, and the record number to access are set, the DISK OUTPUT routine can be called which will write the record to disk. If the DISK INPUT

FILE SPECIFICATION FOR PARAM.DAT

BYTE	DATA NAME	# CHARACTERS
1 - 7	PARAMETER NAME (N$)	7
8 - 10	PARAMETER NUMBER (PN$)	3
11 - 15	UNITS (U$)	5
16 - 21	MINIMUM RANGE (MN$)	6
22 - 27	MAXIMUM RANGE (MX$)	6
28 - 33	ALARM MINIMUM (AL$)	6
34 - 39	ALARM MAXIMUM (AH$)	6
40 - 60	(RESERVED FOR FUTURE USE)	

Table 4, FILE SPECIFICATIONS FOR PARAM.DAT

FILE SPECIFICATION FOR DISPLAY.DAT

BYTE	DATA NAME	# CHARACTERS
1 - 3	POSITION 1 DATA WORD NUMBER	3
4 - 42	PARAMETER VALUES FOR DATA WORD	39
43 - 43	DEFAULT DATA TYPE	1
44 - 46	POSITION 2 DATA WORD NUMBER	3
47 - 85	PARAMETER VALUES FOR DATA WORD	39
86 - 86	DEFAULT DATA TYPE	1
87 - 89	POSITION 3 DATA WORD NUMBER	3
90 - 128	PARAMETER VALUES FOR DATA WORD	39
129 - 129	DEFAULT DATA TYPE	1
130 - 132	POSITION 4 DATA WORD NUMBER	3
133 - 171	PARAMETER VALUES FOR DATA WORD	39
172 - 172	DEFAULT DATA TYPE	1
173 - 175	POSITION 5 DATA WORD NUMBER	3
213 - 214	PARAMETER VALUES FOR DATA WORD	39
215 - 215	DEFAULT DATA TYPE	1
216 - 250	(RESERVED FOR FUTURE USE)	

Table 5. FILE SPECIFICATIONS FOR DISPLAY.DAT

routine is called, the record will be returned in A$. It is up to the calling routines to form or parse the string A$ into its appropriate parts. For this reason, each variable in the system must be the correct length. If they weren't the parsing routines throughout the system would not function correctly. There are other ways to set up a disk structure but this one was chosen because of simplicity and speed. See the Radio Shack book on the Disk Operating System (3).

There are an additional two basic subroutines found throughout the display processor software system. These are titled LINE INPUT and TOGGLE INPUT. Their flowcharts are found in figures 48 and 49 respectively. The purpose for the LINE INPUT subroutine is to input a line of code. A basic INPUT statement from BASIC was insufficient for this purpose. The programmer must pass three parameters to LINE INPUT, a position on the screen at which the data is to be displayed, the number "N" of characters to be typed in, and a default string in A$ to be displayed upon entry. The routine then prints the default string, places brackets around it, creates a flashing cursor to tell the user where the next input will fall, and decodes the control keys for either exit, move within the window, or move to another line. The only area the user is allowed to move around in is the N character space. The flowchart shows the position in the window as "PSN" and prints A$ when it prints the brackets.

The TOGGLE INPUT routine has a similar function but does it in an entirely different fashion. See figure 49 for the flowchart for TOGGLE INPUT. The purpose for the routine is to allow the user to only input a few selected entries. These entries are toggled back and forth using

several control keys shown on the flowchart until the up, down, clear, or enter keys are pressed. When this happens, the selection shown on the screen is returned to the calling program. The programmer calls the routine with the position on the screen to present the display, each value to be toggled between, and the number of them.

Program MAIN's flowchart is shown in figure 50. This simple program displays a master menu and implements a CASE function to perform the selected option. Even though BASIC does not have a CASE statement, its structure is used throughout the display processor software using various BASIC statements. The flowchart is self explanatory.

Program FSETUP's flowchart chart is shown in figure 51. It first initializes some key variables and system defaults, then displays a master menu of selections to choose from. Four inputs are allowed, "L" for load, "E" for exit, "C" for create, and "Q" for quit. The subroutine to convert a record to system variables is necessary because each record is stored as one string as mentioned above.

The record used to store the frame specification is 102 of DISPLAY.DAT. Its length is 250 bytes which allows plenty of room for the 76 bytes needed to specify the frame data. The structure of the record is shown in Table 6 below. Once the record is read in from disk, it is broken out into variables shown in the table. The main subroutine in FSETUP is EDIT which is called by three options, "L", "E", and "C". Its flowchart is shown in figures 52 and 53.

The EDIT subroutine is essentially one large CASE function. The FSETUP entry page consists of 10 lines, each line requires special

considerations for data entry. The flowchart shows each consideration for each line number. Note the extensive use of the LINE INPUT and TOGGLE INPUT subroutines. For those lines which require error checking, the flowchart shows a return to the beginning of the column. For example in line 1, no more than 16 bits are allowed for the wordlength. Therefore if the entry is larger than 16, the user is returned to the beginning. Line 1 also creates a pad of blank characters to be used for the sync word definition string in lines 8, 9, and 10. To work, each entry must be the correct number of characters long for the disk access procedures discussed above. For the sync word entry, the number is 16. If the wordlength was 10 bits, then a pad of 6 would have to be added to the sync entry to make it 16 characters long.

Part 2 of the flowchart shows the return sequence form the pseudo case structure. The line number is incremented and checked against the

FRAME SPECIFICATION LAYOUT IN RECORD 102 OF DISPLAY.DAT

BYTE	DATA NAME	# CHARACTERS
1 - 2	BITS PER WORD (B$)	2
3 - 5	WORDS PER FRAME (W$)	3
6 - 13	PCM TYPE (TY$)	8
14 - 17	PCM BIT RATE (RA$)	4
18 - 20	PCM ORDER (O$)	3
21 - 27	PCM POLARITY (PO$)	7
28 - 28	NUMBER OF SYNC WORDS (SW)	1
29 - 44	SYNC WORD 1 (SW1)	16
45 - 60	SYNC WORD 2 (SW2)	16
61 - 76	SYNC WORD 3 (SW3)	16
77 - 250	(NOT USED)	174

Table 6. FRAME SPECIFICATION FILE STRUCTURE

limits 0 and 10. If they are reached, wraparound occurs which places the line pointer to either the top or bottom line number. This is common practice and used throughout the display processor system software.

The PSETUP program flowchart is shown in figure 54. Again the case structure is used extensively with several new subroutines, LIST, and RETRIEVE DATA.

The LIST routine is shown in figure 55 and does several functions. When called by a program, its purpose is to display a listing of the parameters in the data base PARAM.DAT. A simple loop would do fine if the display page were long enough but only ten lines of data are allowed at one time; if ten are present, the program gives the user the option of continuing, selecting a number, or exiting. To do this function, it first outputs the basic list page without any data. It then reads the maximum number of records from record 201 in the PARAM.DAT FILE. This number becomes the variable MAX shown in the flowchart. The loop reads and displays ten records in at one time unless it gets to MAX first. It then gives the user the options mentioned above. If "E" is chosen, the return variable A$ is 0 and the calling program will take this as no entry. If a number is selected, (one of the numbers on the listing page), that number will be returned to the calling routine to be processed. In this case, the calling routine was EDIT so the selected parameter will be displayed on the EDIT PAGE. If any other key is pressed, the listing will continue.

The RETRIEVE subroutine shown in figures 56 and 57 presents a display page which interacts with the user for data input. This

routine is very similar to the EDIT routine used in FSETUP and acts the same way. Its structure is shown in the figure and implements many small CASE structure. After each line of data is input using the LINE INPUT subroutine explained earlier, the users exit key is decoded to either backup one line, move down one line, or quit. The wraparound mode is implemented a different way than the one in the EDIT subroutine was but it works just as well. The user stays in the loop, editing a line at a time until the quit command is given. One major difference in the RETRIEVE subroutine from EDIT is that it has the capability of storing data before returning to the calling program. Upon the quit command, the alarm entries are checked against the maximum and minimum entries to detect a conflict. If there is one, the user is returned to where the conflict occurred. If there isn't an error, another case type structure gives the user the option of saving the file, returning to the major line entry loop, or returning to the calling program. A trick used throughout the display processor system is to use the BASIC statement RUN. This has the nice feature of clearing all variables by starting the program from scratch, and saving a few lines of code. It is only used when a RETURN statement would do the same thing.

Now that the major subroutines are explained, the structure of PSETUP will make a little more sense. PSETUP, as mentioned earlier, is a 6 element case structure each exiting with a RUN statement. The ADD option reads in the max number of records stored and adds one to it. This will be the new record number. Next the RETRIEVE subroutine is called. If the returned key is "T" for Try again, the user is placed back into the RETRIEVE routine. If an "S" is returned, the RETRIEVE

subroutine has already saved the data so the only thing left to do is up the max record value stored in record #201.

The "E" function gives the user the options to list the data base, enter a number or exit gracefully. If a number is entered, it is checked against the limits of the system and either read in or aborted. The record is then read in and dumped to the RETRIEVE routine as shown.

The next two functions require moving records around on disk and thus take quite a bit of time to do. The sub-options of "NUMB", "L", and "E", are the same as in the last option discussed. For "INSERT", the old record is read into a temporary variable so that room can be made for a new one. The RETRIEVE subroutine is called with blank data because of the RUN command explained earlier. The user puts in new data and returns from RETRIEVE as before. If the "SAVE" option was used, the new record will now be on the disk instead of the record previously saved. The remainder of the code moves all records above the saved record up one record starting from the top and working down. This will leave a space for the record saved in the temporary variable.

"DELETE" works just the same way except that the file that the use chose to delete is brought into the DISPLAY PAGE using the DISPLAY DATA subroutine. DISPLAY DATA simply prints out the data to the screen without any tricks. It does call for a user key input which is acted upon when returned. IF the key is "Y" as shown, all records are moved down one record starting at the record just above the one deleted. If "Y" is not selected, the command is aborted. The final option form the PSETUP MAIN MENU is "QUIT". This simply runs the program MAIN.

The program DSETUP is structured almost exactly like program

PSETUP described above. The only differences are the data file and an added option from the master menu. See figure 58 for the flowchart.

The data file is titled DISPLAY.DAT and structured as shown in table 5. DSETUP is structured around ten display lines per page but each record can only store the data from five parameters. This requires the use of two records per display page. The flowchart shows only reading one record in at a time but this actually means read one display page in at a time. In actuality, two physical records are read in for each logical record. A physical record is the actual record as it is written on disk. For DISPLAY.DAT, an arbitrary maximum number of data records is set to 100. Since each display page (logical record) requires two physical records, only 50 display pages can be in the system at one time. A small calculation converts the logical records to two physical records and converts the data therein to variables to be used by the program. See the listing.

The additional option which can be called from the master menu, is the "LIST DISPLAY PAGE" feature. The user has the option of either listing the parameter (data word) database, or the display page database. A separate flowchart is shown in figure 59 for this subroutine. Its structure is exactly the same as that for the "LIST DATA WORDS" routine except each logical record read is actually two disk records as explained above.

The EXECUTE program flowchart is shown in figures 60 and 61 with special subroutine flowcharts in figures 62 thru 64. The main flow of control is shown in figure 60. The structure is complicated by the fact in that the main routine is not a master menu as it was in all

other programs in the system. There is a master menu of course, but it is called as a subroutine from the engineering display function. This simplifies data flow because all control functions such as changing data types, displaying bar graphs and, plotting data words are based around the engineering display page. To the user, the structure is exactly the same as other programs in the system but to the programmer, it's different.

The program first initializes some basic variables as is done in all other programs. Next the master menu subroutine is called to allow the user to see the status page, display engineering data as it is read in, or quit. The routine for showing the status is simple as all it does is print a few variables in a specific format on the screen. Any key will return the user to the master menu. The quit function will run program MAIN as all other programs do. When the user wishes to see engineering data, a RETURN is issued and the program continues.

The program then displays the raw display format and headers calls the LINE INPUT routine shown as read line in the flowchart. The user must input a page number to be displayed before the program continues. It then reads the appropriate logical record corresponding to the page number selected and prints the static information to the page. Static data are those items that aren't updated such as the parameter name, parameter units, data type etc. Next, the major update loop reads the data from the memory buffer and displays it to the user.

The major loop is shown in figures 60 and 61. The synchronization status is checked first by reading the status flag in the I/O control block (table 3). This is done before any information is displayed to

prevent wrong data from entering the system. If the status flag is true (not zero), the system is synchronized with the PCM data stream and the program continues. If not, it waits for the status to become true, or a user key input of "E" for exit. If the status becomes true, one data value is read in from the I/O memory buffer, converted to engineering units if its type is decimal, or converted to its non-decimal format as raw data and displayed. The program then checks to see if there is a user command and if not, reverts back to the status check and the loop continues.

Figure 61 shows the case structure of the key decode routines. Nine options are available as shown, one option for each column of the flowchart. If the keys are left arrow, right arrow, or a number, the loop is exited and a new page is brought in to be displayed. Wrap around mode is implemented if the user elects to use arrows. If the up and down arrows are used, the selected line shown by an inverse data type indicator will be changed, either up or down. The algorithm uses a selected line pointer, variable LM in the listing, and just increment and decrements it using wraparound techniques. toggle type and plot options use LM to identify the data word to perform their respective operation on.

The toggle type function uses a routine much like the TOGGLE INPUT routine used within the system but doesn't expect a user input. It just changes the type identifier in the selected line's type column by reading the screen location directly using PEEK and POKE statements. The main loop reads this type and does the respective conversion. The toggle function also replaces the units column on the selected line to

the data type if not decimal. If it is, the units identifier is displayed.

The next three figures, 62 - 64, show the flow diagrams for the plotting functions. Figure 62 is a routine called by the BAR CHART and PLOT GRAPH routines which plots a point on the text screen using special text-graphic characters. By using these special characters, the lengthy and time consuming graphics capabilities of the Color Computer are avoided which makes the routine easy to modify. The routine is quite simple but tricky because it has to select the appropriate character to print. The characters used are blocks instead of alphanumerics. There are many types of blocks, each with their own number. The number is an extension of the ASCII character set and ranges from 128 to 255. Only two blocks are used in the plot routine, they are numbers 147 and 156. The number 147 prints an inverse block one character width wide and one half character tall at the bottom of the character space. Number 156 does the same but places the block at the top of the character space. In this way, 20 points can be plotted with only 10 lines available on the screen.

The plot routine is called with an X and Y value, X being the column (0-31) to print the character block and Y being a percentage in the range 0-99. The routine converts these values to an absolute number beginning with 0 to 511 which represents the character location on the screen. Zero is the upper left character position and 511 is the lower right. The numbers increase from left to right and top to bottom much like that of the scanning of a TV picture. The conversion is simple because of the in-line nature of character position

numbering. Since Y is in the range from Ø to 99, the first thing to do is convert it to a vertical position from Ø to 19. If the position is even, the bottom block, number 147, should be used. If odd, 156 should be used. To determine this, the value is divided by 2 and checked to see if it is an integer or not. An integer will denote an even number so 147 is selected. If not, 156 is chosen. To offset to the correct line number and allow room at the bottom of the page for prompts etc., an offset of 3 is added. The resultant Y value is the line number counting from the bottom of the screen where the chosen character is to be placed. Since the leftmost character position on the screen is the line times 32 counting from the top, the actual position on the left side of the screen is 480 - (Y * 32). Adding X to this gives the correct location on the screen to place the selected character. Before printing the new block character, however, the old one must be erased. This is done by the remaining code in the routine using a double buffering method. The flowchart shows how this is done.

The BAR CHART routine flowchart is shown in figure 63. The routine is a misnomer in that bars are not actually shown except for the alarm limits. Only a point is plotted to show the value in percent of full scale. The routine is basically a major loop which plots ten data values at one time against a bar showing the alarm range. The first thing the routine does is draw the chart display and static data much like the engineering display routine did. It next draws in the alarm limits in bar format by plotting the maximum alarm point, the minimum alarm point, then all points in between. If the alarm limits are zero, the routine is bypassed. Once the pointers are initialized,

the routine waits for sync using the same method used before. If in sync, it reads in the data, converts it to an X and Y value, and then plots it. Any keypress will return control back to the engineering section because there is no user interaction for BAR CHART plotting. If a key wasn't pressed, the data word pointer is incremented using wraparound and the loop continues.

The PLOT CHART routine of program EXECUTE has two modes of operation, continuous and trace. The flowchart appears in figure 64. After converting the word count read from the I/O CONTROL BLOCK to a sample time in microseconds, (time = word count/16), it displays it on the screen, Then the type of mode is checked and the appropriate path taken.

Continuous mode is much like that of the bar chart option except that 29 different values of the same data word are plotted instead of 10 different data words. If, at any time, the user presses "T", the mode is changed to trace and the trace path is taken. See the flowchart. If "E" is pressed, the program reverts back to the engineering display page. Any other key will keep plotting in the continuous mode.

The trace mode is a little more complicated in that synchronization between the PCM processor and the display processor is now important. After setting the PCM processor in trace mode by placing a data word value in the valid flag of the I/O control block, the PCM processor begins a trace at the beginning of the memory buffer. If the data rate is slow, it could take some time to fill so the program must check to see if data is ready to be read by looking at the

data address word in the I/O control block. If the returned address from the PCM processor is less than the address the display processor is ready to access, the system checks for a user halt and, unless received, continues waiting for the addresses to match. Once the data address is larger, the data is valid and the program displays it and checks for the end of page. When at the end of the page, 29 points have been plotted. The buffer can hold 1024 points so there is more data to be displayed. To do this, the user must use the left and right arrows to scan the memory buffer and plot points accordingly, one page at a time. If at this point, the exit option is invoked, the PCM processor is returned to continuous mode and the user is placed back into the engineering display page. If CONTINUOUS is selected, the mode flag and PCM processor are reverted back to continuous operation.

The flowcharts for the short machine language CONVERT routines are not shown but described in the listing.

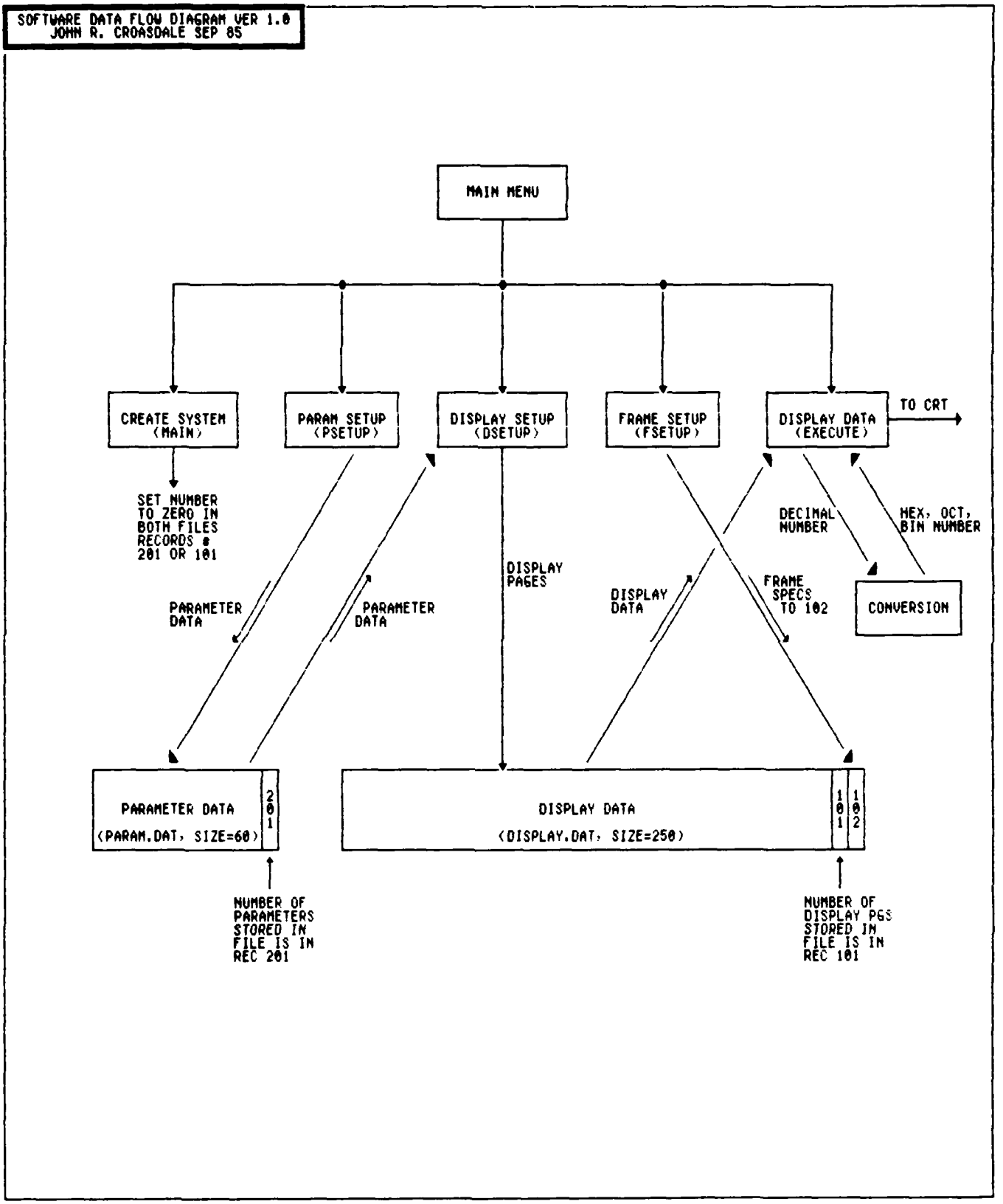


Figure 47. Software Data Flow Diagram

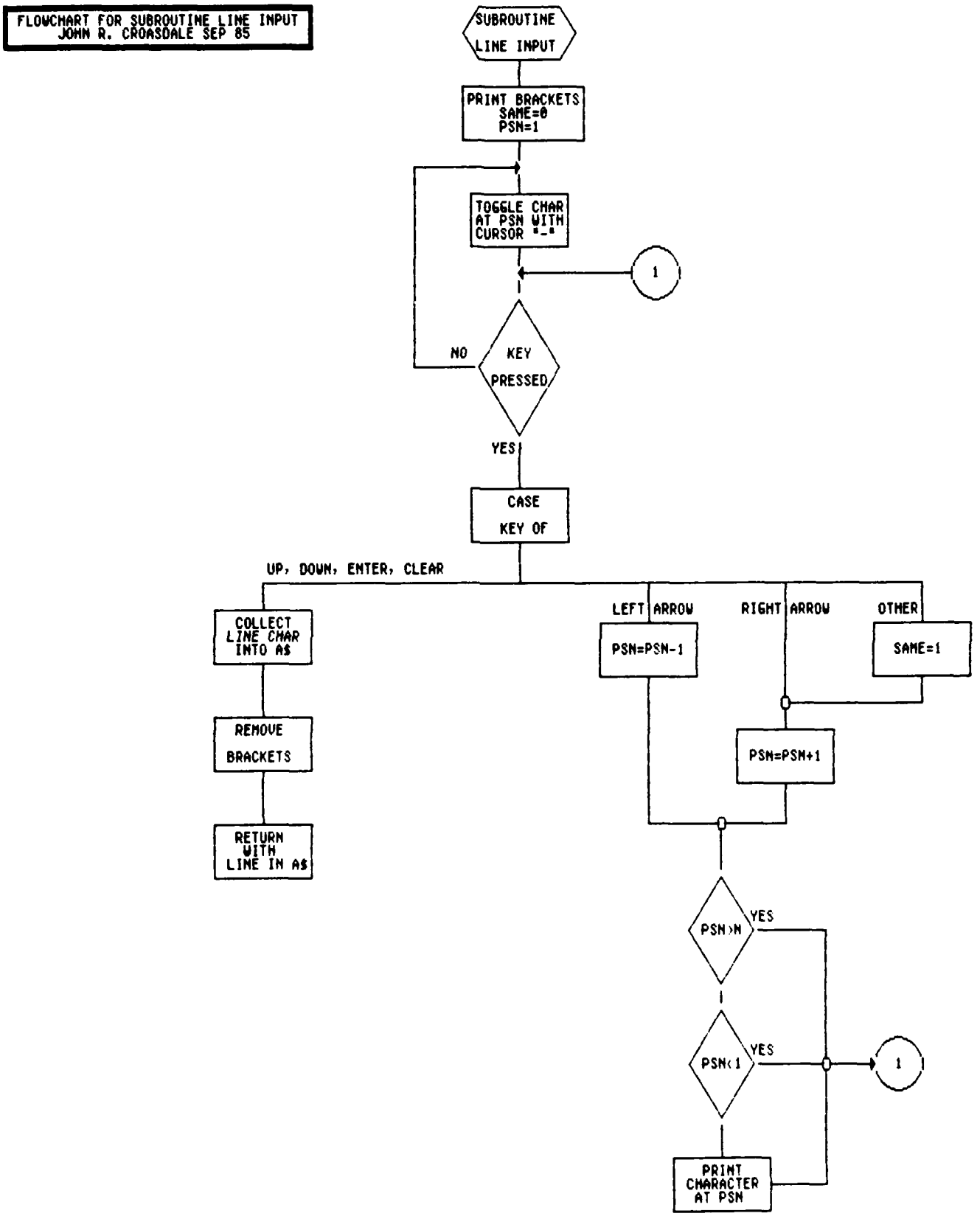


Figure 48. Line Input Flowchart

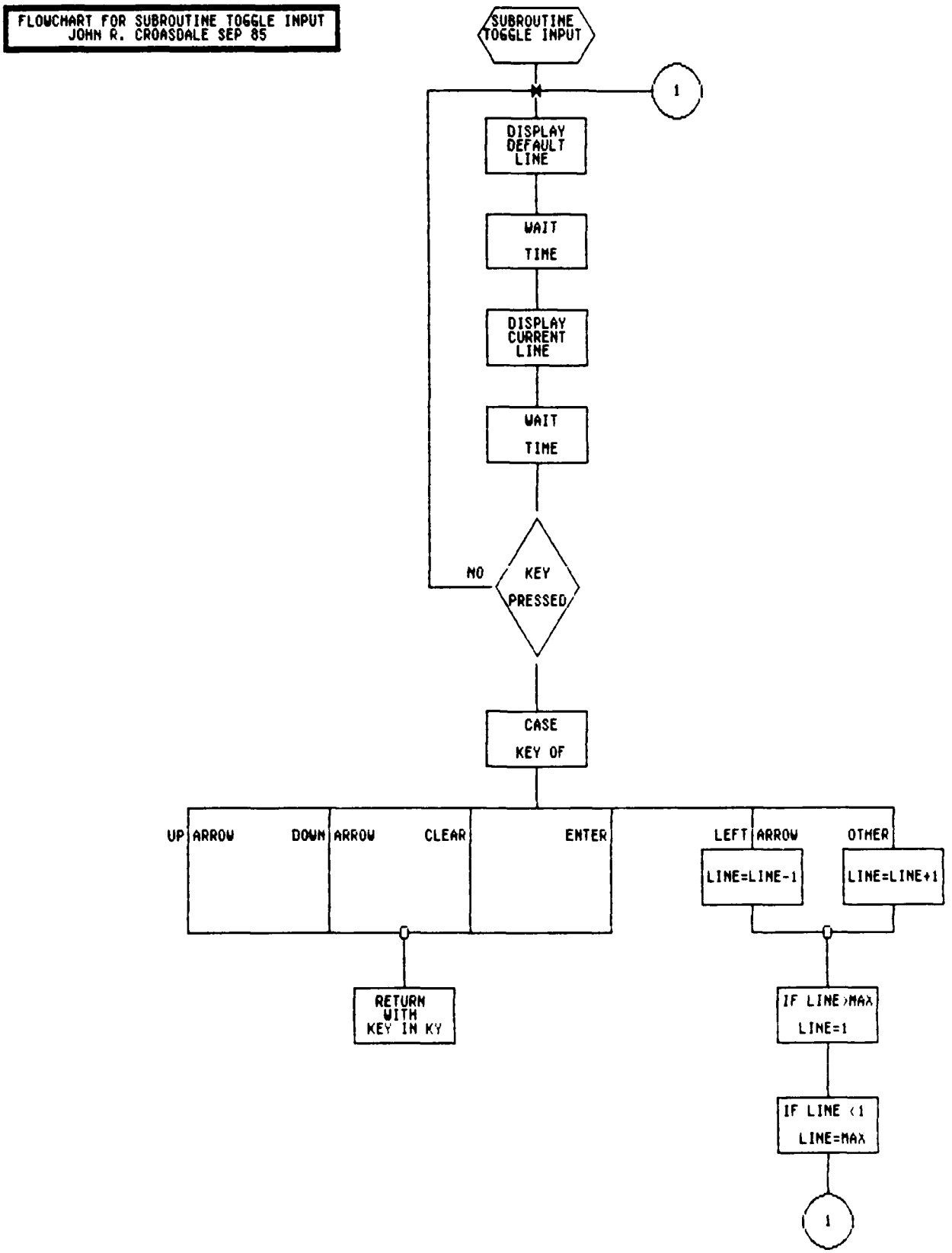


Figure 49. Toggle Input Flowchart

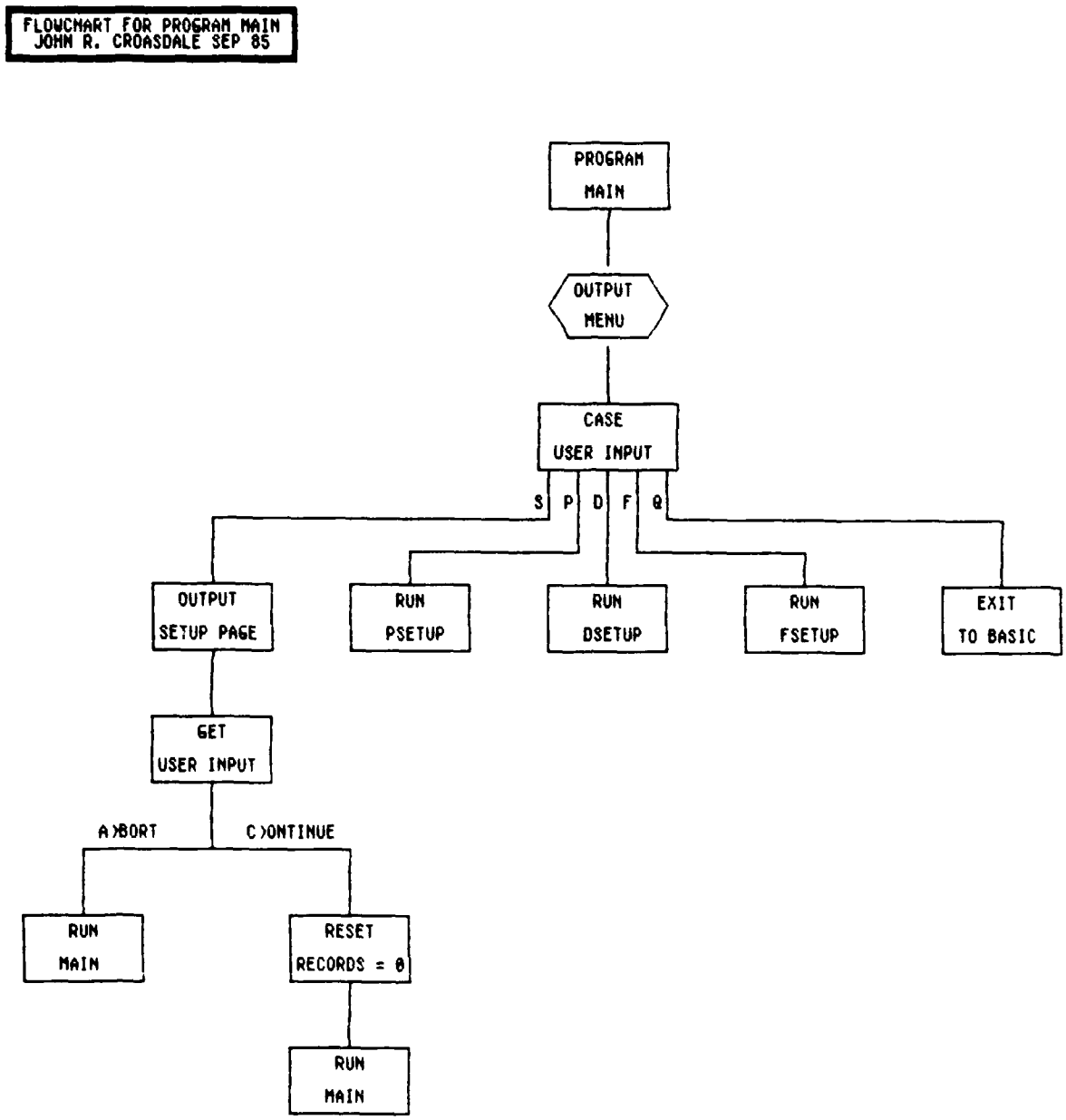


Figure 50. Program MAIN Flowchart

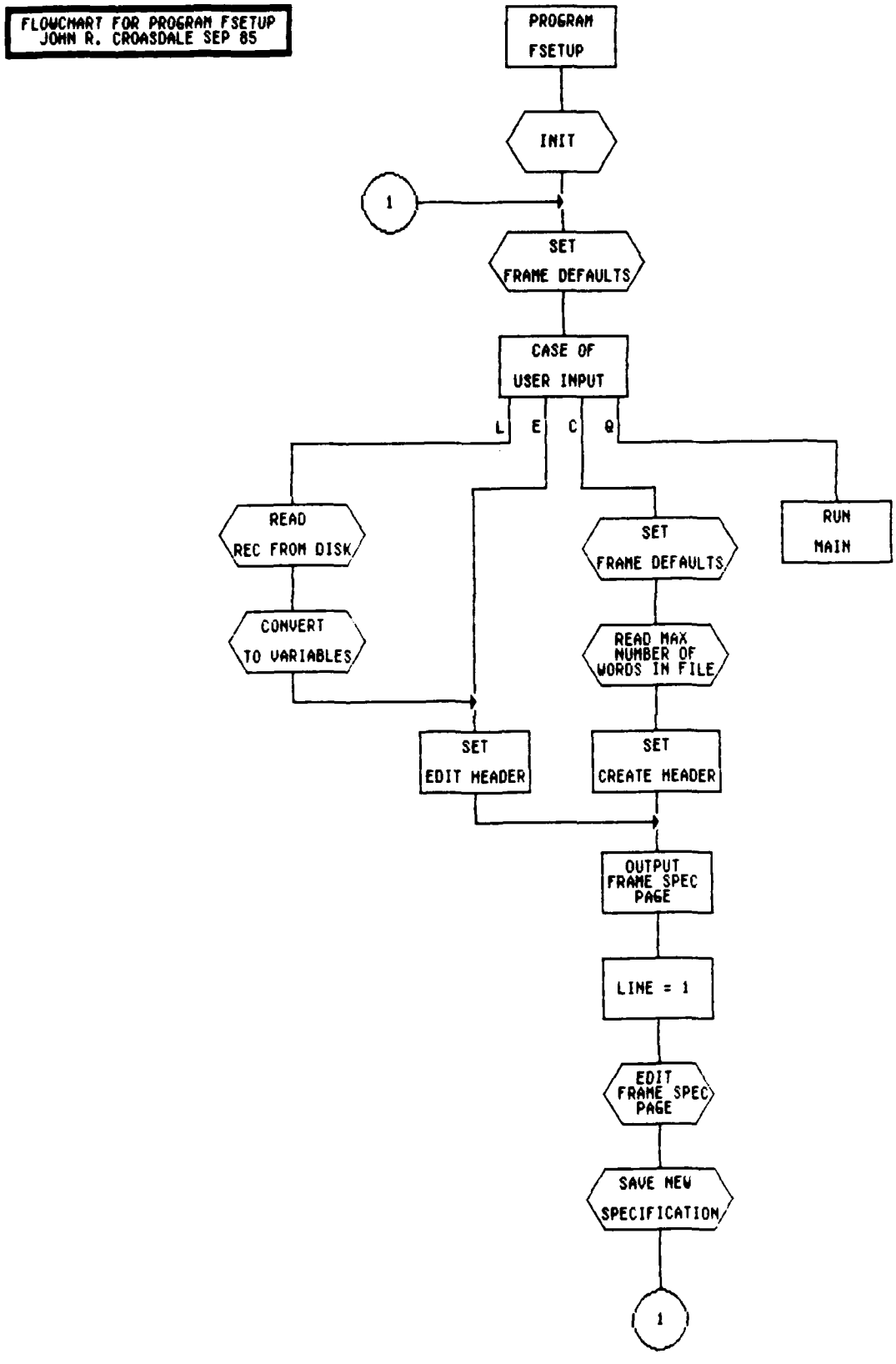


Figure 51. Program FSETUP Flowchart

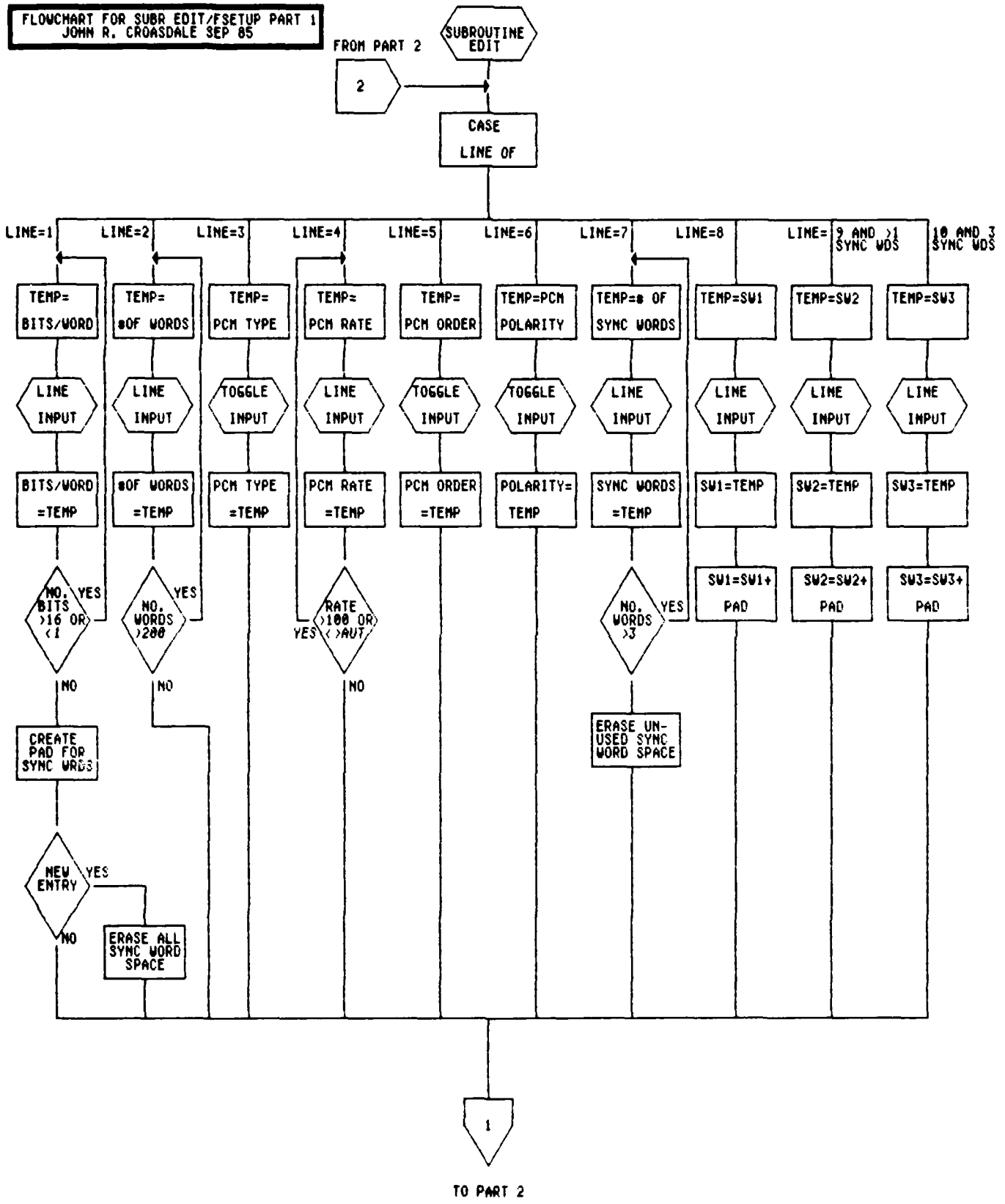


Figure 52. EDIT Flowchart Part 1

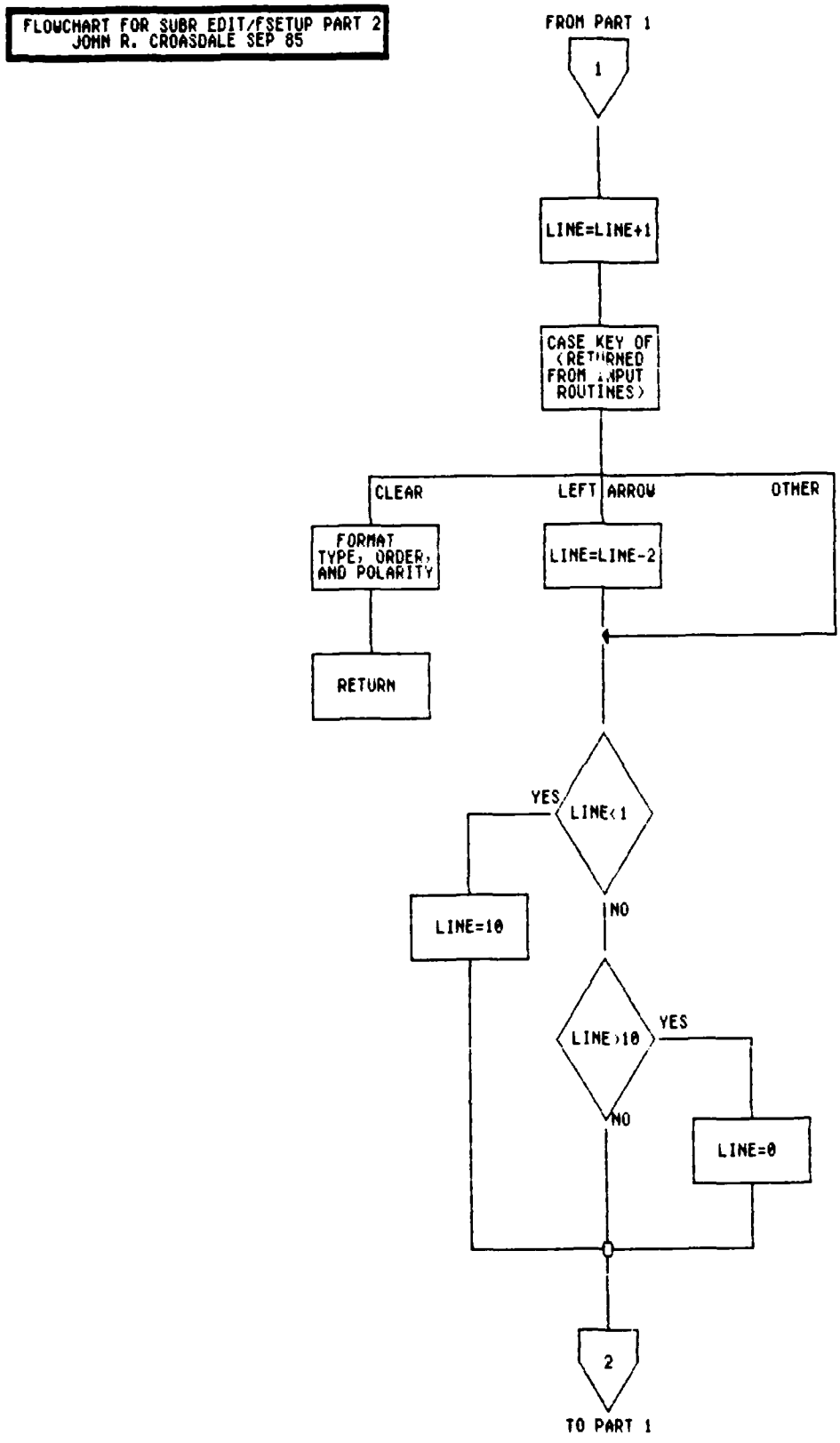


Figure 53. EDIT Flowchart Part 2

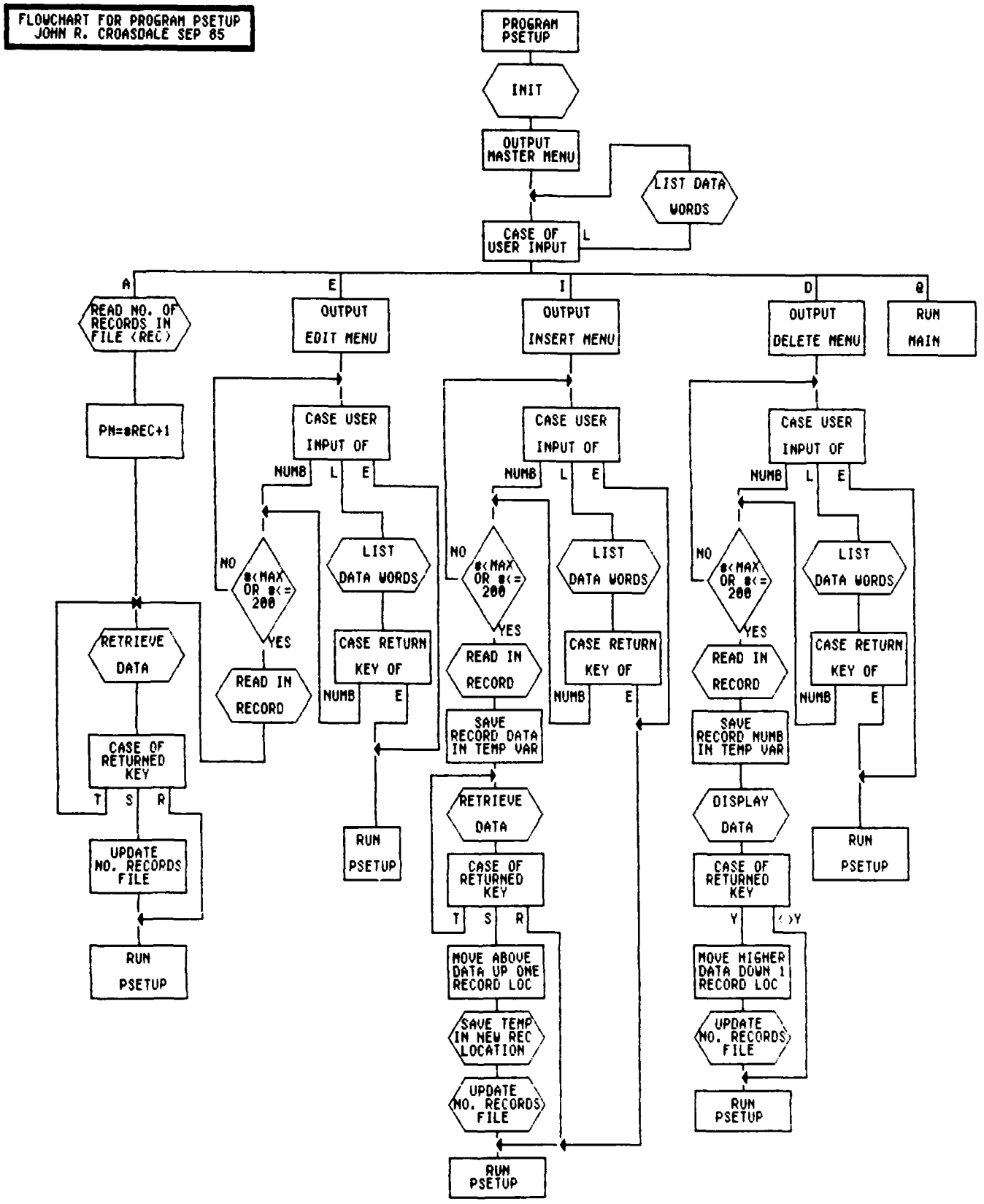


Figure 54. Program PSETUP Flowchart

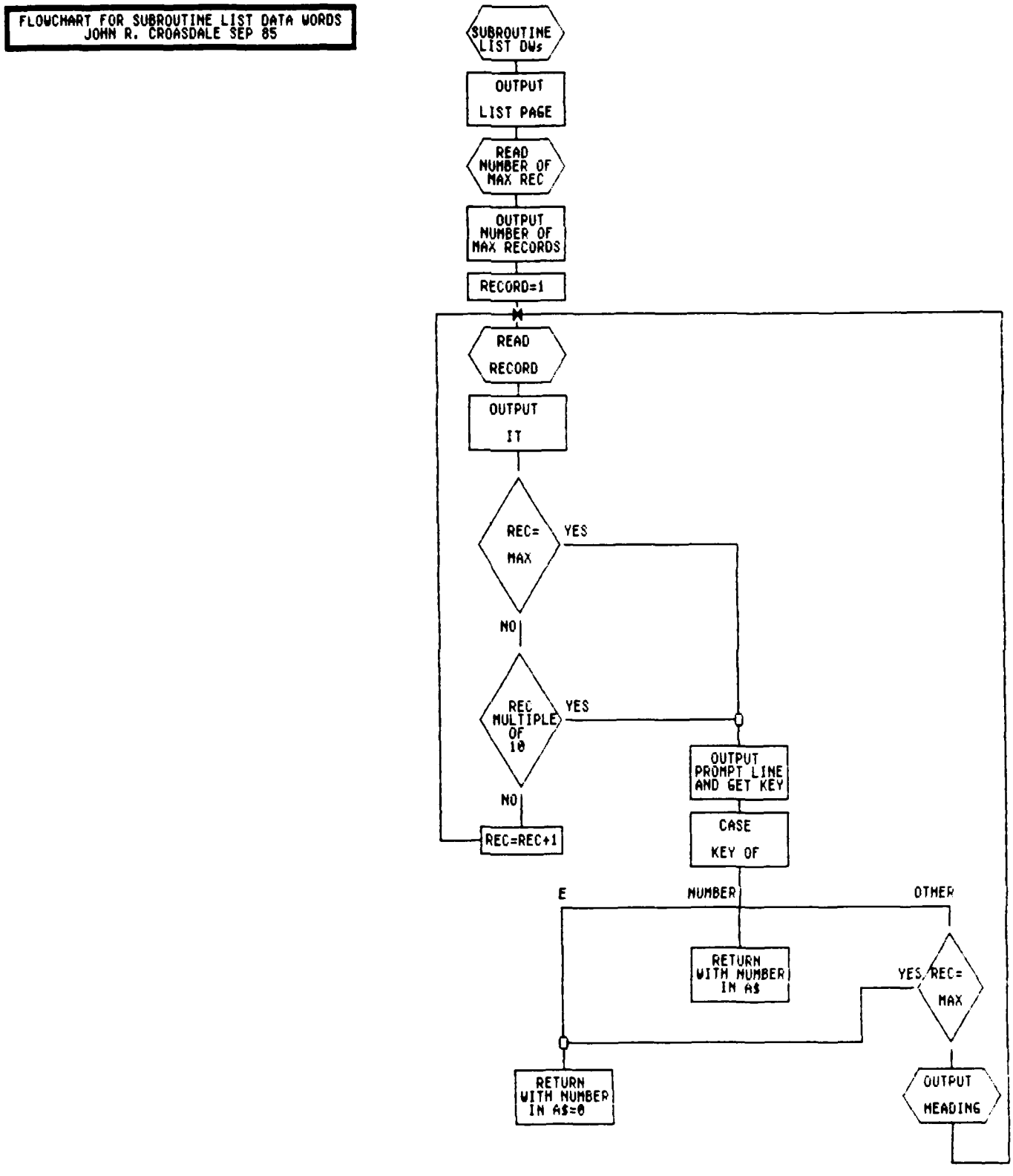


Figure 55. LIST DATA WORDS Flowchart

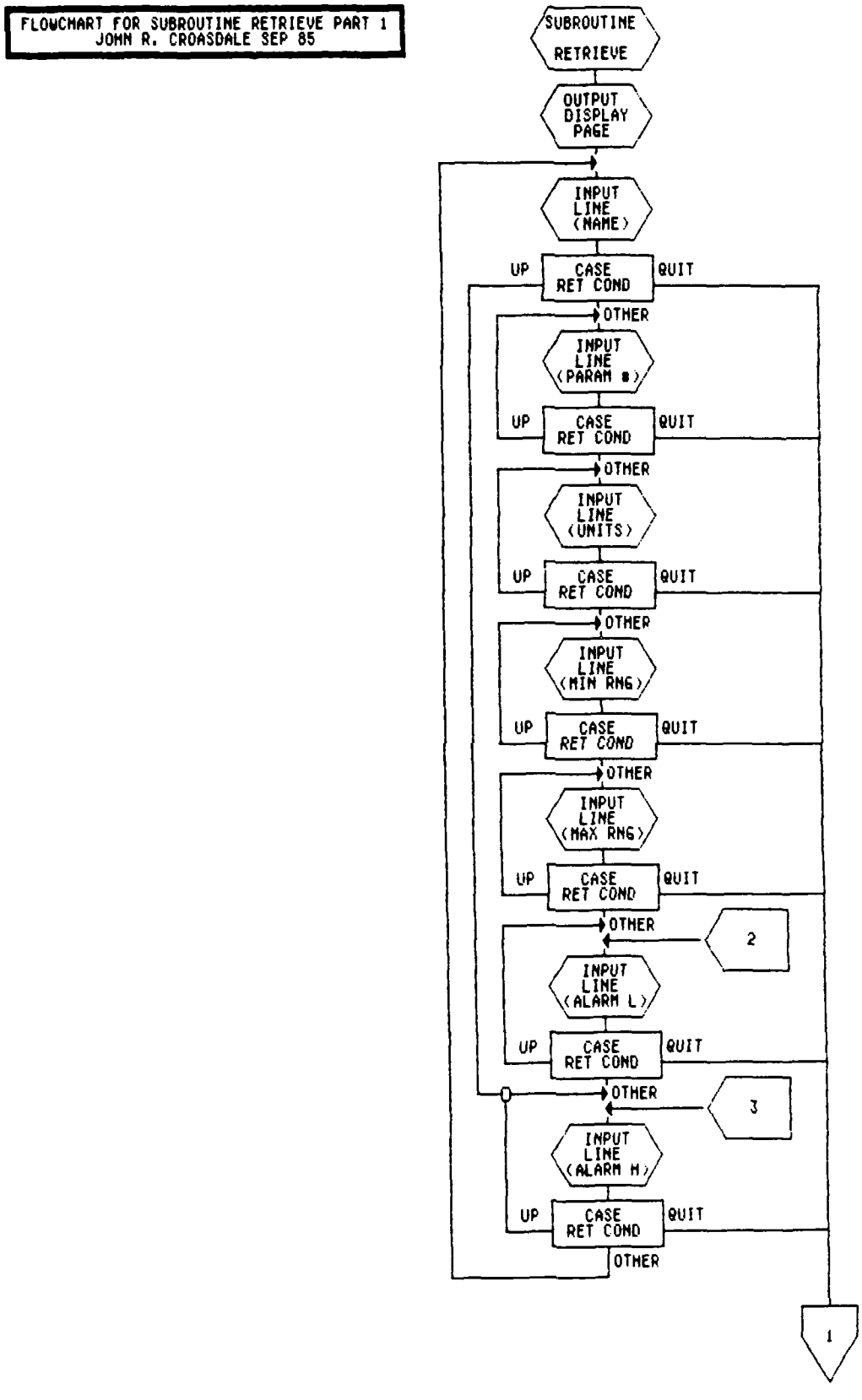


Figure 56. RETRIEVE Flowchart Part 1

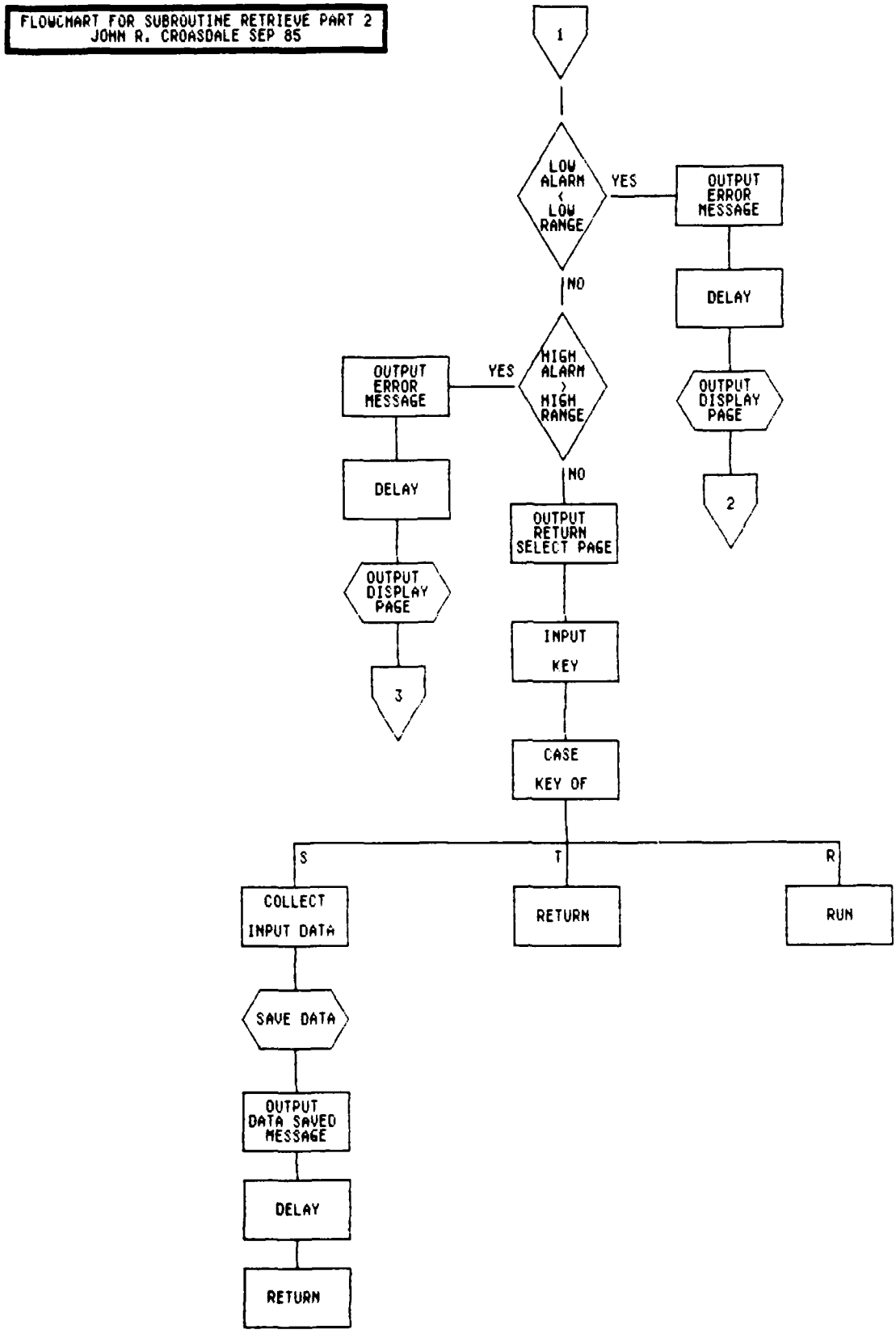


Figure 57. RETRIEVE Flowchart Part 2

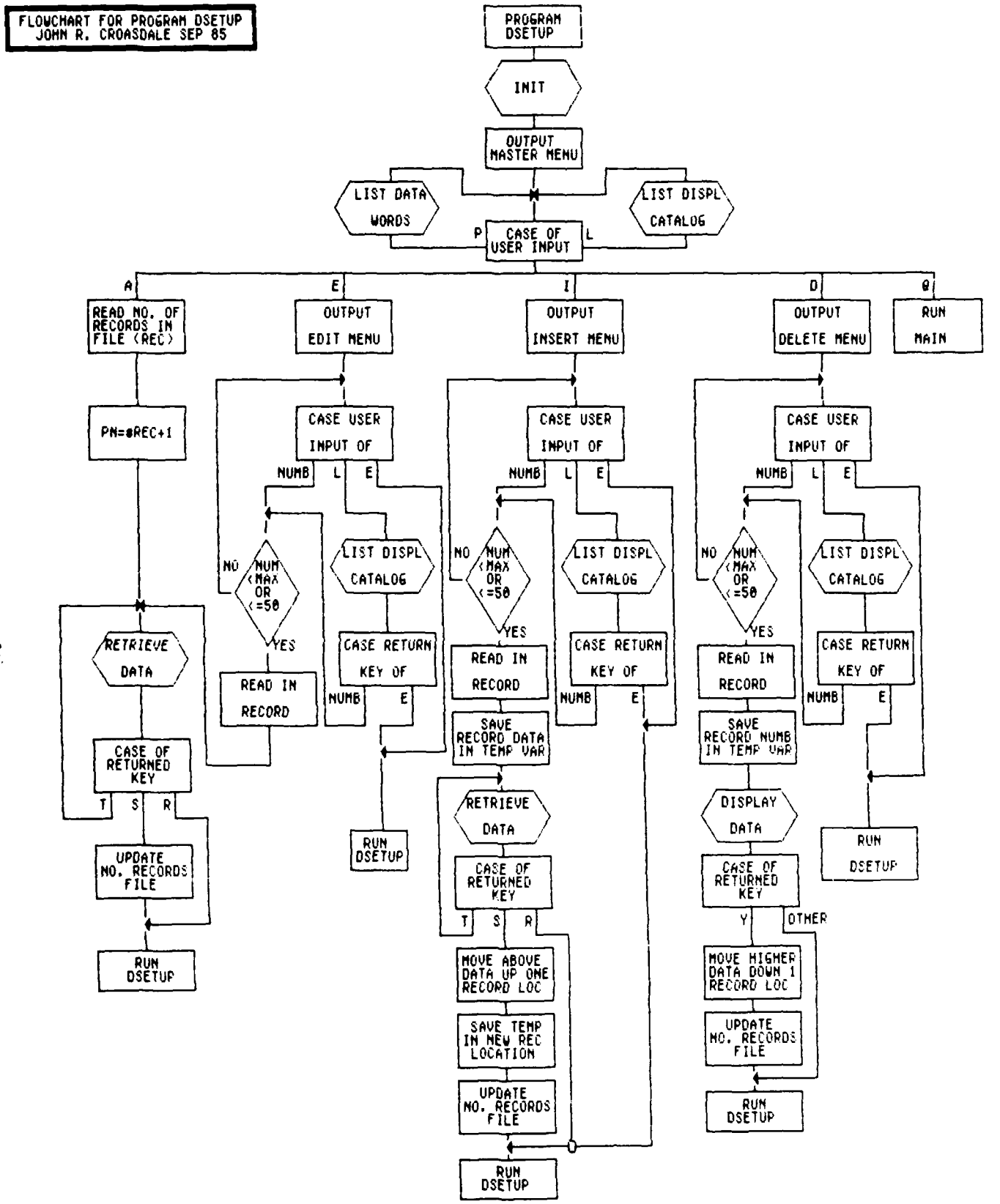


Figure 58. Program DSETUP Flowchart

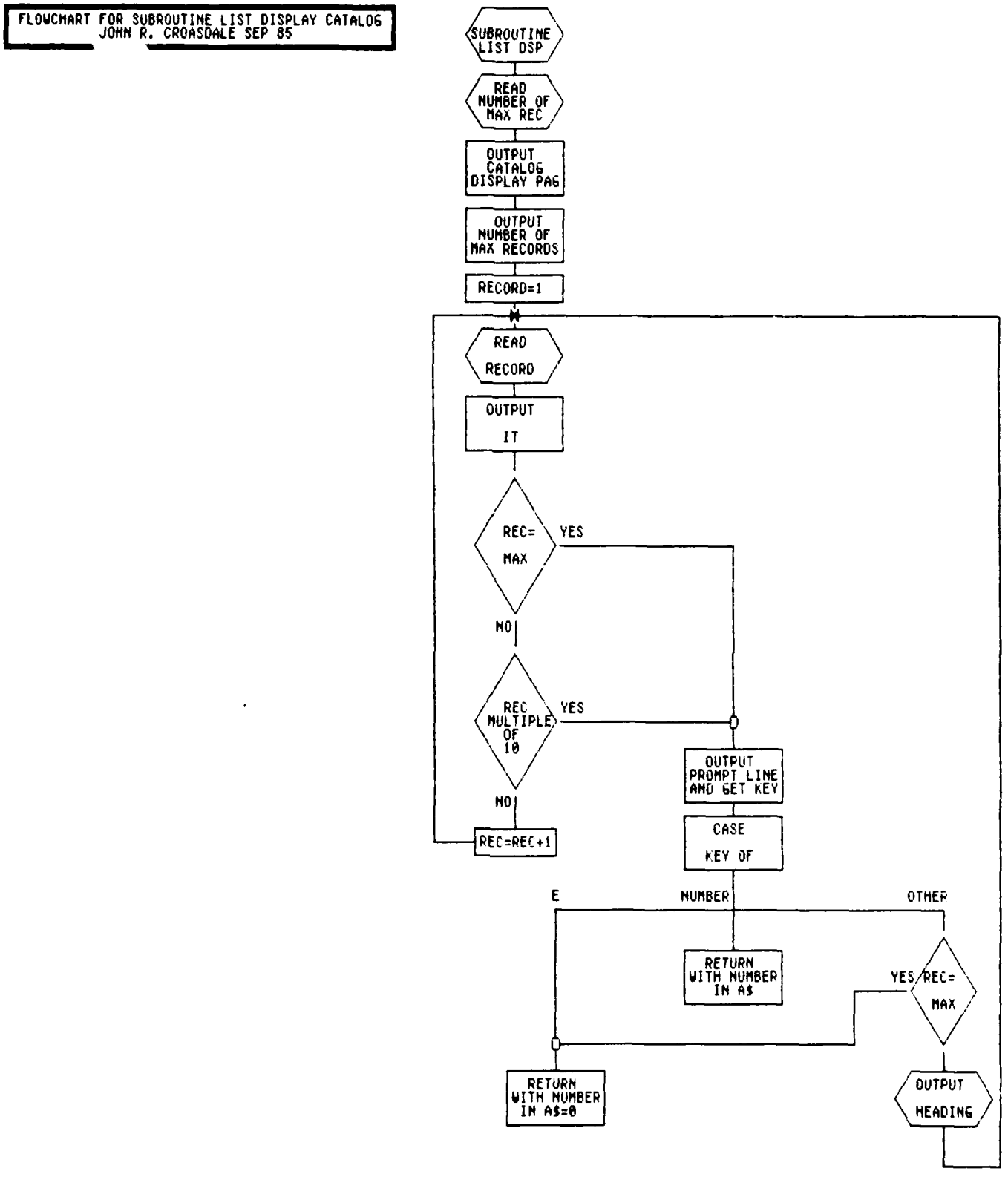


Figure 59. LIST DISPLAY CATALOG Flowchart

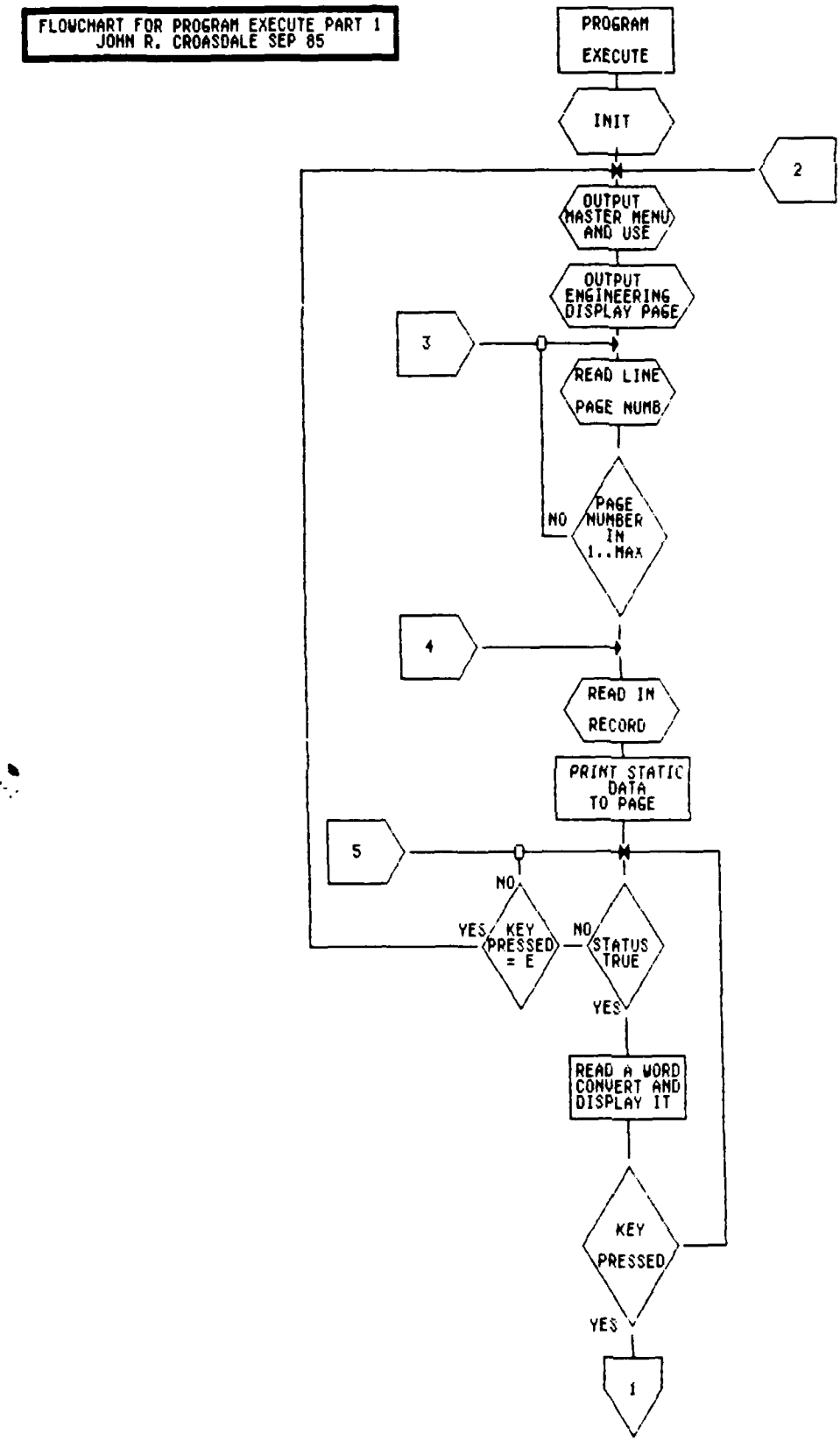


Figure 60. Program EXECUTE Flowchart Part 1

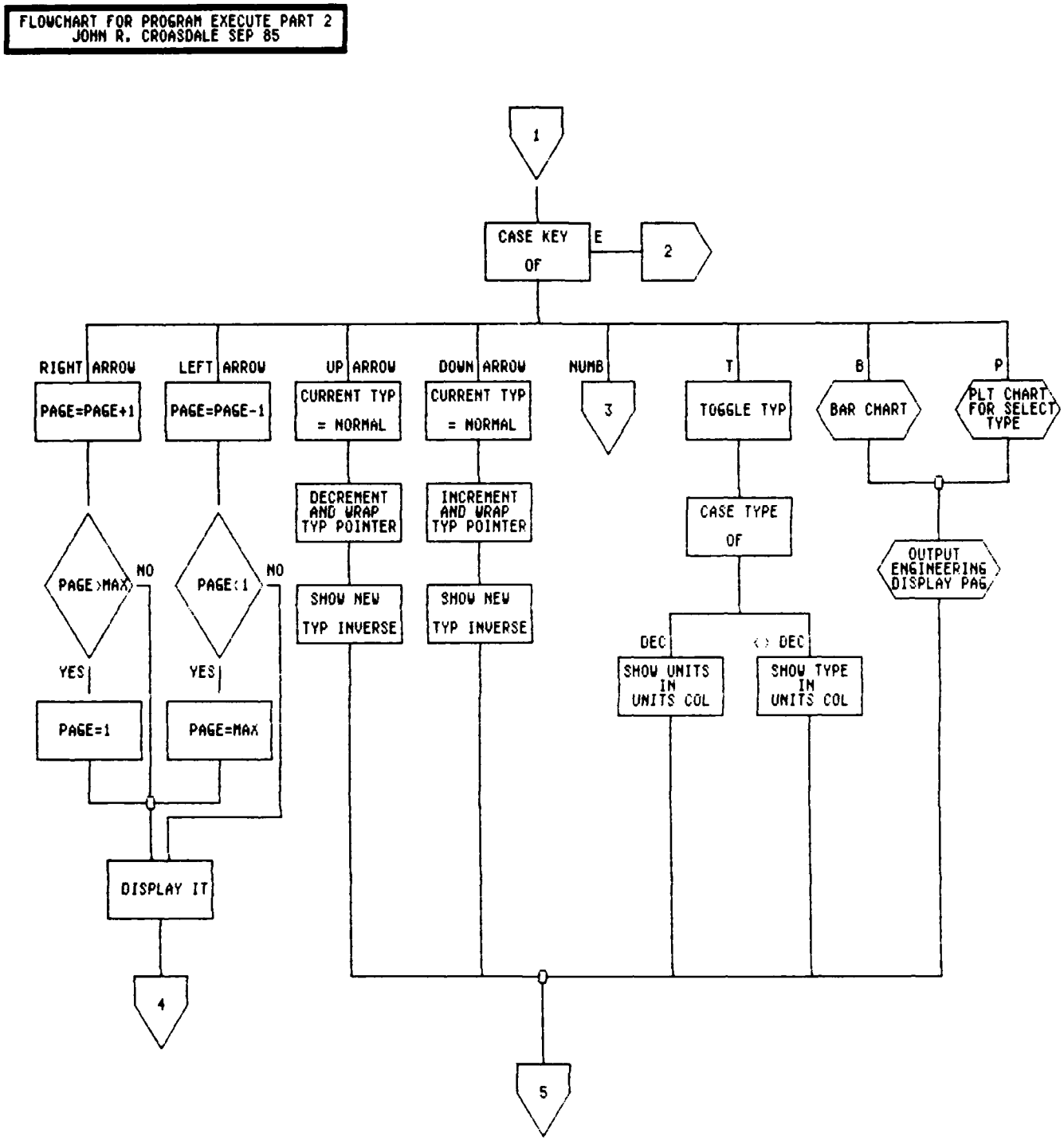


Figure 61. Program EXECUTE Flowchart Part 2

FLOWCHART FOR SUBROUTINE PLOT POINT
JOHN R. CROASDALE SEP 85

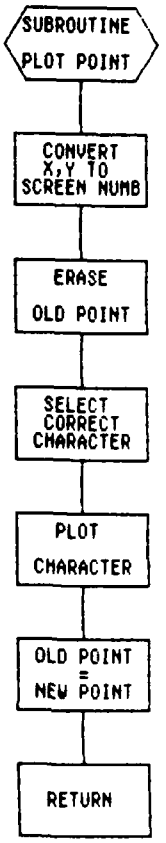


Figure 62. PLOT POINT Flowchart

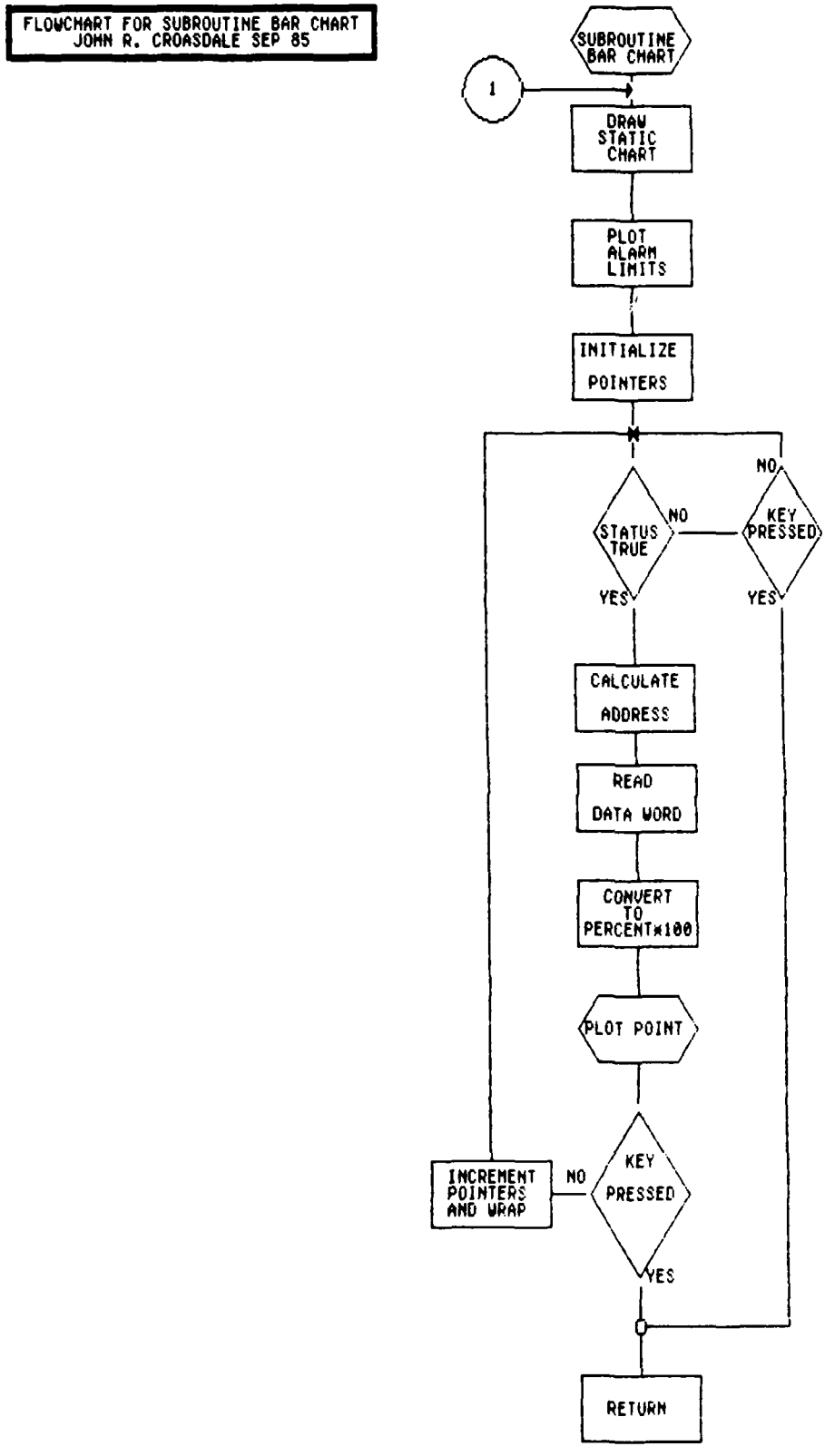


Figure 63. BAR CHART Flowchart

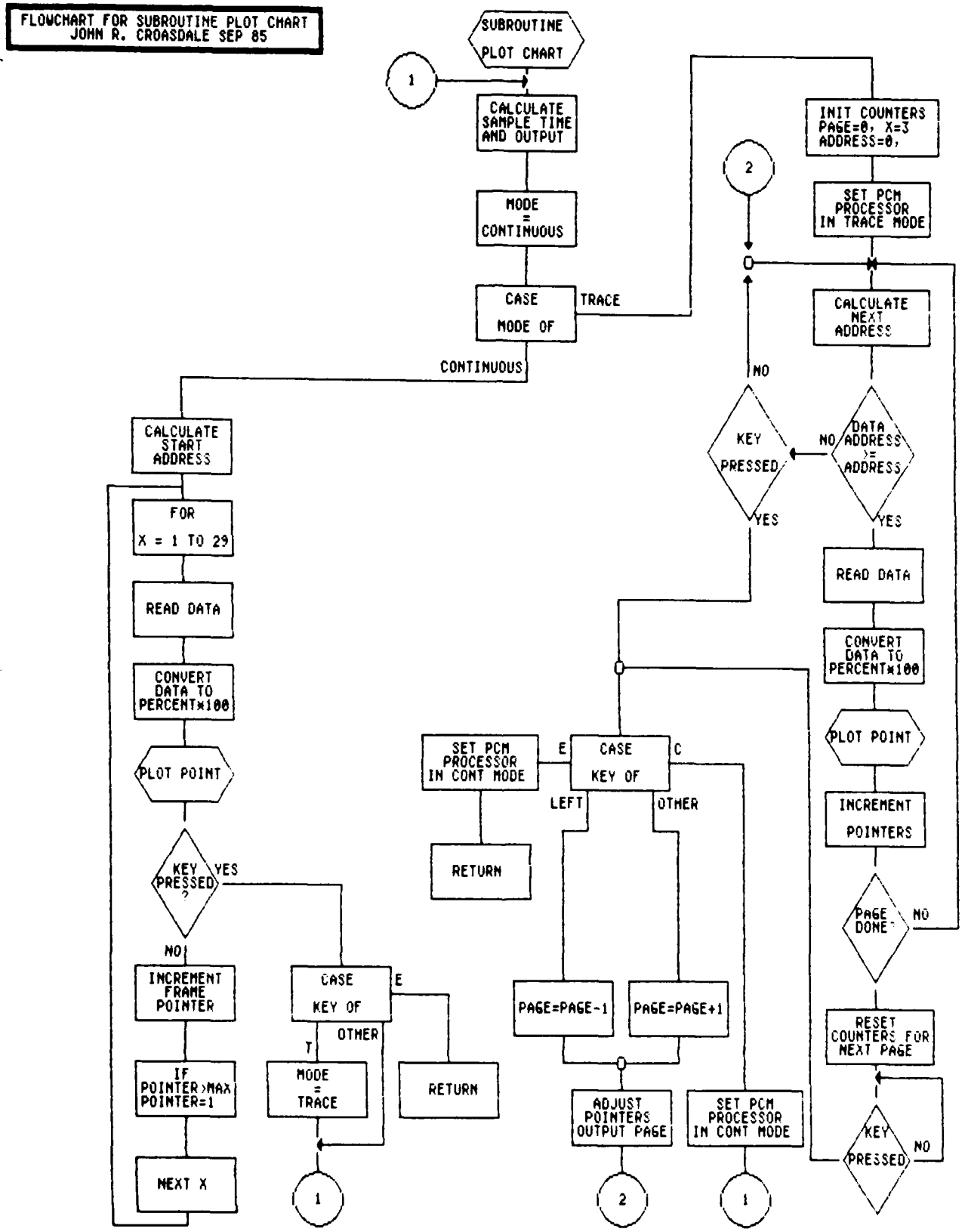


Figure 64. PLOT CHART Flowchart

References Used in This Manual

1. Croasdale John R. Real-Time Flight Test PCM Acquisition Monitor. AFIT Thesis 85S-1.

2. Kaine, Gerry, Doug Hawkins, and Lance Leventhal, 68000 Assembly Language Programming; Berkeley, CA, OSBORNE/McGraw Hill, 1981.

3. Leventhal, Lance A. 6809 Assembly Language Programming; Berkeley, CA, OSBORNE/McGraw Hill, 1981.

4. Motorola, 16 Bit Microprocessor User's Manual, Englewood Cliffs, N.J., Prentice-Hall, 1979.

5. Getting Started With Extended color Basic, Tandy corporation, FortWorth, Texas, 1984.

6. Radio Shack TRS80 Color Computer Disk System Owner's Manual and Programming Guide, Tandy corporation, FortWorth, Texas, 1981.

Appendix B

REAL-TIME FLIGHT TEST PCM

DATA ACQUISITION MONITOR

USER'S MANUAL

BY

JOHN R. CROASDALE, LTC, USAF

Appendix B

User's Manual

Overview

This manual provides the basic step-by-step procedures needed to operate the Real Time Flight Test Data Acquisition Monitor, PCM monitor for short. It first provides a basic description of the system followed by a walk through of setting up and using the PCM monitor system in a basic test. The main thesis describes the system in more detail and should be consulted if more information about a particular function is needed. The user is assumed to know about PCM transmission protocols.

System Description

The PCM monitor consists of two major parts, the PCM processor, and the display processor. The PCM processor is the interface circuitry and software needed to convert a PCM data stream to a series of data words which can be read by the display processor. There is no user interaction with the PCM processor except the PCM order switch and the manual reset explained below. All other interaction is through the display processor.

The display processor is a host computer system with software which controls the PCM processor. The user interacts with the display processor by entering PCM data word and frame specifications, and then displaying selected information in various formats.

The software in the display processor is completely menu driven

with the exception of a few items. These items will be explained as they arise in the discussion below. The basic overview of the software command structure is shown in figure 65 and should be reviewed before proceeding with this section.

Throughout this discussion, instructions within quotes must be typed exactly as shown. Instructions enclosed within apostrophes are Color Computer keys which should be pressed. Any time a command such as "L)IST" is seen on the menu, only the first character needs to be typed followed by the 'ENTER' key to invoke invoke it. Some commands do not require the "ENTER" key to be pressed such as the arrows and toggle type functions described below.

Getting Started

The PCM monitor is presently housed in two sections. One section is the PCM processor card with power supply and the other is a Radio Shack Color Computer II with disk drive and video monitor. Connect the PCM signal cable to the PCM processor's input jack and make sure the ribbon cable from the Color Computer is connected to the PCM processor card. The cable must be extending away from the processor board and not across it. Turn on the power to all systems: computer, disk drive, and PCM processor card. Check the power supply for the PCM processor card for a reading of about 1.5 amperes. If it exceeds a value of two, turn the system off immediately and check for short circuits. A standard 5 volt power supply capable of delivering 2 amps is required for the PCM processor card.

Make sure the system diskette is inserted in drive Ø. The BASIC

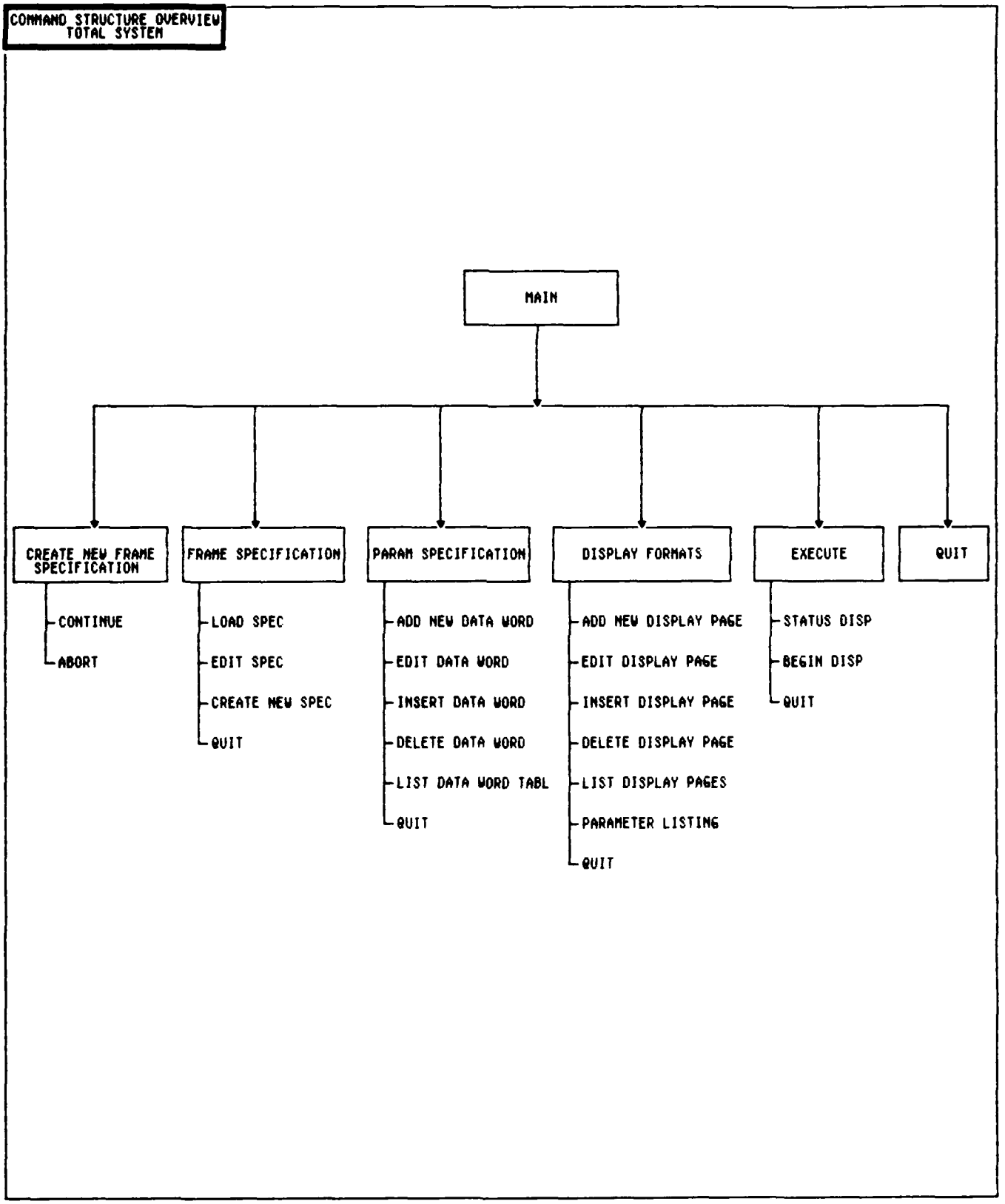


Figure 65. Command Structure Overview

prompt will be displayed on the monitor when the Color Computer is turned on. To start operations, program MAIN must be run by typing "RUN MAIN" then the 'ENTER' key. The disk drive should come on and after a few seconds, the display shown in figure 66 will be displayed. The user is given six options to choose from.

For new systems, the first option should be "S)ETUP NEW SYSTEM", selected by pressing "S" followed by 'ENTER'. The page shown in figure 67 will be displayed. The user responds by typing a "C" for continue or "A" for abort. Since the old data will be replaced with new data, the old data base, PSETUP.DAT and DISPLAY.DAT, must be saved under a new filename before entering the PCM monitor system. By pressing "A" and then "Q" for quit, the user will be placed into BASIC for the filename change. If the "C" key is pressed from the setup page, the program will continue and set up the data files for a new system.

Old data systems and newly created ones can be added to by using the other options. Since this example assumes a new data system, all following discussion will be from the new system viewpoint.

Setting up Frame Specifications

Once a new system has been initialized, the frame specification should be entered. Pressing "F" from the MAIN master menu, bring up the FSETUP master menu shown in figure 68 below. Since this is the first time through the system, the "C)REATE NEW FRAME SPECIFICATION" option should be invoked by pressing the "C" key followed by the 'ENTER' key. The other two options, "E)DIT" and "L)OAD" are used to modify previously entered data from either memory using the "E)DIT"

```
**********************************
*                                *
*           MAIN MENU            *
*                                *
**********************************

  F)RAME SPECIFICATION
  P)ARAMETER SPECIFICATIONS
  D)ISPLAY FORMATS
  X)ECUTE TO BEGIN PCM MONITOR
  S)ETUP NEW SYSTEM
  Q)UIT

   WHICH ONE-->?
```

Figure 66. MAIN MASTER MENU

```
**********************************
*                                *
*       SETUP NEW SYSTEM         *
*                                *
**********************************

WARNING WARNING  WARNING WARNING

THIS PROGRAM WILL DELETE YOUR
DISPLAY FILES.  BE SURE YOU'VE
SAVED THEM TO ANOTHER FILENAME

DO YOU WISH TO CONTINUE OR ABORT
PRESS C OR A ->?
```

Program MAIN has a master menu page shown in the top of this figure. If the S function is called, the lower page will be displayed. If the user responds with a "C", the program will set up a new system data base. Any other key will revert back to page 1.

Figure 67. SETUP MENU

option or the disk file using the "L)OAD" option. Only one frame specification can be active at one time. All three options bring up the data entry page shown in figure 69.

The frame specification page in figure 69 shows an example of a typical PCM frame specification. When running the program, the "CREATE" function will display the default values. These values are shown as a number or in inverse text as required. There are no defaults for the "SYNC DEF" values so this area of the page is left blank (not shown in figure 69). There are seven items which need to entered to describe the frame format. The arrow keys are used to move up and down the seven line entries and to scroll left and right within a line.

Upon entry, the cursor is placed on the first line, "BITS/WORD". Here the user types a number from 1 to 16. The normal size of a PCM word is around 10 to 12 bits but for greater resolution, 16 bits can be used. The number of bits, as well as all the other entries on this page, is dictated by the PCM data stream format and not the PCM monitor. The user must use what is in the PCM stream.

The second line entry is "WORDS/FRAME". Any number of words are allowed up to a total of 200. "PCM/TYPE" is either NRZ-L or BI-PHASE-L. A third option on this line is "AUTO". The PCM monitor has the capability of determining whether the signal is in NRZ format or bi-phase. Selecting "AUTO" allows the system to use the calculated type.

Perhaps the most important innovation of the PCM processor is its capability to capture a PCM bit rate from 1Khz to 100Khz automatically.

```
**********************************
*   FRAME SPECIFICATION PAGE     *
**********************************

C)REATE NEW FRAME SPECIFICATION
E)DIT SPECIFICATION IN MEMORY
L)OAD FRAME SPECIFICATION
Q)UIT

WHICH ONE->?
```

Figure 68. FSETUP MASTER MENU

```
**********************************
*   FRAME SPECIFICATION PAGE     *
**********************************

BITS/WORD        10  MAX IS 16
WORDS/FRAME      34  MAX IS 200
PCM TYPE         AUTO NRZ BI-PHASE
PCM BIT RATE     AUTO KHZ A)UTO
PCM ORDER        LSB MSB
PCM POLARITY     NORMAL INVERSE
# OF SYNC WORDS  3  MAX # IS 3
SYNC DEF->  1111101011
            1100110011
            0100000000
**********************************
 USE NUMBERS, ARROWS, OR CLEAR
```

Figure 69. FRAME SPECIFICATION PAGE

The user has the option of manually setting a bit rate by typing a number in the "PCM BIT RATE" line. Such a number could be 25.5 meaning 25.5 thousand bits per second. By typing the letter "A", the "AUTO" feature is selected. If a number is used, the PCM processor will use this number but it will return the calculated value to the user. The calculated value will be displayed on the status page of the EXECUTE program explained later.

"PCM ORDER" can be either Most Significant Bit (MSB) first, or Least Significant Bit (LSB) first. The order is determined by the PCM encoder. Generally, PCM systems send data LSB first, then the next to least etc. It then gets the next word and sends it out in the same manner. If the PCM monitor is not told what this order is, the data may become transposed and be meaningless. The order is switch selectable on the PCM processor card.

The "PCM POLARITY" option is used to switch from a high level meaning "1" to a high level meaning "Ø" etc. It is normally used for bi-phase as some systems use a transition from low to high to mean "Ø" and high to low to mean "1". Other systems use the reverse. The polarity function is accomplished in the PCM processor software to allow for any type of encoder polarity outputs. Consult the thesis'software section on PCM processor for more information on polarity.

The number of synchronization words is entered on the next line. The word length for the sync word must be the same as the data word length. Up to 3 sync words can be used. The "SYNC DEFINITION" lines follow to allow the user to enter the binary representation of the

word. If only one sync word is allowed, the program will allow only one to be entered. If two are allowed, only two can be entered. The same for three. The program also checks the word size from line one and allows the user to enter only those number of bits. If the "BITS/WORD" is changed for some reason, the "SYNC DEFINITION" display area is erased and must be re-defined. This keeps the user from entering erroneous data

Pressing the 'CLEAR' key at any time during the data entry process saves the data displayed shown on the page to disk. If items are left blank, the default options shown in inverse are used. There is no way to abort this page without saving the displayed data to disk unless the 'BREAK' key is pressed. If this happens, the system should be re-started by running MAIN. All data entered, however, is still in contact and need not be re-entered.

Setting up Parameter Files

To set up the parameter data base, select the "P)ARAMETER SPECIFICATIONS" options from the MAIN master menu. The data word definition page shown in figure 70 will be displayed. The user is offered the options to "A)DD" new words, "E)DIT" words already in the system, "I)NSERT" data words between others, "D)ELETE" data words, "L)IST" all the data entered in the system, or "Q)UIT". Press the "A" key to add a new data word and the system will bring up the data entry page placing the cursor in the first data entry line. Notice that the system automatically calculates the data word number and displays it on the screen. In this case, the data word being entered is 1.

A brief explanation is in order here to distinguish between a parameter and a data word. Parameters are those data words in the PCM stream which are to be displayed and used to determine if the system under test is functioning. Data words are numbered according to their locations in a frame with data word number 1 being the first and data word N being the last; N being the number of data words in the frame. Parameter number 1, for example, might be cabin altitude represented by the 4th data word in the frame. In this case, the data word number would be 4 while the parameter number would be 1.

The flashing cursor designates the position for text entry. In this case, the system is asking for a parameter name. Up to 7 characters may be entered in this line. The brackets show the limits. Both letters and numbers may be entered to make parameter identification unambiguous. The limits were needed because of other page limits in the PCM Monitor system due to the limited display capability of the Color Computer screen. Type in the first data word name, "ALTITUD" and 'ENTER'. The cursor will move to the next line and ask for a parameter number.

In this example, altitude is item number 4 on the parameter list so press "4" and 'ENTER'. Actually the parameter number does not have to be a number. The user can use this entry as some other type identifier as needed such as "ACF" showing that it is an aircraft parameter rather than a system one. Its intended use, however, is to help the user to keep track of his parameter listings.

The next entry is the units of the parameter. Dimensionless data will have no units but in this case, altitude is measured in feet so

```
**********************************
*                                *
*   DATA WORD DEFINITION PAGE    *
*                                *
**********************************

A)DD NEW DATA WORD
E)DIT DATA WORD
I)NSERT DATA WORD
D)ELETE DATA WORD
L)IST PARAMETER TABLE
Q)UIT

WHICH ONE?
```

Figure 70. PSETUP MASTER MENU

```
   DATA WORD DEFINITION PAGE
**********************************
ADDING DATA WORD # 1

PARAMETER NAME --> ALTITUD
PARAMETER NUMBER-> 4
PARAMETER UNITS -> FEET

RANGE MINIMUM -> -100
RANGE MAXIMUM -> 50000
ALARM MINIMUM -> 200
ALARM MAXIMUM -> 45000

**********************************
ENTER FOR NEXT, CLEAR TO ACCEPT
```

Figure 71. A)DD NEW DATA PAGE

type "FEET" 'ENTER'.

The range minimum and range maximum relate to the full range of the expected output of the transducer supplying the data to the PCM encoder. These are engineering or real-life units to which the engineering display will be calibrated to. For example, when the digital data for this data word is zero, (all bits are Ø), the user should expect to see a certain value on the display which is set up during the transducer calibration. In this case, a zero reading would be scaled to -1ØØ feet. Unsigned numbers are assumed to be positive. If the data words were 16 bits long, the maximum absolute data read into the system would be 65535 or all bits set high (1). The system would then be calibrated to give a maximum reading of, in this case, 5ØØØØ feet. In no way could the system display values lesser or greater than "RANGE MINIMUM" or "RANGE MAXIMUM" respectively. The absolute readings from the 16 data bits in this example would then be mapped or scaled to fall within the range high and range low limits and be presented to the user.

Alarms are set to alert the user if the data is exceeding a preset value. In this case, the user wants to know when he/she is getting close to the ground so "ALARM LOW" was set to 1ØØ feet. In addition, an "ALARM HIGH" was set to 45ØØØ feet to alert the user when the aircraft is above a save limit. Alarms are displayed only on the bar chart display explained later. These useful prompts can tell the user of imminent aircraft damage due to an altitude of 1ØØ feet or lower

At the bottom of the page are several command prompts. The "ENTER" keys on the Color Computer is used to jump from one line to the

next. In addition, the arrow keys are also used to move within a line, or from one line to another. Pressing the "CLEAR" key brings up the accept page shown below in figure 72.

The user can either "SAVE" the data just entered to disk and return to the master menu, "TRY" again to change data just entered, or "RETURN" to the master menu without saving the data. Data is saved in the PARAM.DAT file in the record synonymous with the data word number and in a format explained in appendix A.

Error checking is done throughout to prevent the user from entering data which would bomb the system at a later time. For example, if an alarm were set outside the range limits, the user would be alerted and returned to the alarm entry position on the add data word page. Once saved, the user will be placed back in the master menu page.

The "E)DIT DATA WORD" option allows the user to edit a previously

```
S)AVE DATA WORD # 1
T)RY AGAIN
R)ETURN TO MENU

WHICH ONE ->?
```

Figure 72. Save Page

- -

defined parameter. When this option is selected, the user is asked which data word he/she wishes to edit. At this point, a number will bring in the specified data word to a page exactly like the add page described above. The user can then make changes to the parameter entry and save as before. Several options are available at this point other than editing a data word. These options are "EXIT" and "LIST".

"EXIT" returns the user to the master menu. "LIST" presents a display shown below in figure 74 which is a summary of all the parameters entered in the system. Since only one parameter was entered, only one will be listed. This "LIST" command is exactly the same as the "L)IST" function on the master menu with the exception that pressing a number will bring up that data word into the edit data word page. If the "LIST" function is brought up from the master menu, pressing a number will only return the user to the master menu as the system does not know what to do with it, edit, delete, insert or whatever?.

Listing a data base helps the user get his/her bearings if the data word number for a particular parameter can't be remembered. It is also very useful in checking to see if all parameters are entered correctly. Of course, the characteristics of the parameters are not displayed but at least its position in the data stream can be verified.

The user has three options from the listing page, all of which are effective when called from EDIT. Pressing a number will bring the selected parameter into the edit parameter part of the program. The user can edit at will and save as mentioned above. A carriage return, or "ENTER" key on the Color Computer, will continue the listing if more

```
ENTER DATA WORD TO EDIT ->?


PRESS L TO LIST, E TO EXIT
```

Figure 73. Command prompts from EDIT function

- -

```
DATA WORD LISTING OF 1   RECORDS
**********************************
     WORD #       NAME  PARM

       1         ALTITUD  4



NUMBER=SEL, CR=CONT, E=EXIT ?
```

Figure 74. LISTING DISPLAY

than ten data words are entered into the system. The listing displays only ten lines at a time to allow the user time to see them. Pressing the "E" key exits the listing and returns the user to the master menu.

The "I)NSERT DATA WORD" option allows the user to insert data word specifications between other data words. This feature is very useful if changes to the parameter list or sequence of the data word order have been made for some reason. Selecting this function brings up a user prompt page similar to that which was brought up if the "EDIT DATA WORD" were selected. The user has the same options, "ENTER" a data word number, "LIST", or "EXIT". If a data word number is entered, the insert data word page will be displayed. This page is exactly the same as the add data word page except for the title. The new data word will be inserted before the number selected by the user. The user can abort at any time but if he/she chooses the "SAVE" option, the data base will be re-ordered to reflect the inserted word. This operation is entirely disk based so for large data bases, the time to complete the operation could take up to 15 minutes. For this reason, the user is continually advised of the disk operations performed during the operation. He/she cannot abort once the operation is underway.

The "D)ELETE DATA WORD" option allows the user to delete a data word. The file is then re-arranged to assure sequential data word numbers. This feature is exactly the same as the "INSERT DATA WORD" option except that the user is shown the data word he/she has chosen in detail on the same page format as the add or edit data word page. At this time, the user is re-asked if the data word is to be deleted. If yes is chosen, the file is re-arranged in the same process as the

"INSERT" option.

The "L)IST PARAMETER TABLE" option does exactly the same function as the "LIST" option in in the edit, insert, and delete modules. There is one exception and that is if a data word is selected from the listing, the program returns to the master menu instead of displaying the selected word. This option is intended to be used to double check all data entries before moving on to the "DISPLAY FORMAT" option from the MAIN program.

The final option is "Q)UIT". Calling this function returns the user to the MAIN module and allows him to call other programs in the system.

Defining Display Pages

Once all of the desired parameters are set up the next step is to define which of them are to be displayed and in which order. Any combination of up to ten parameters may be displayed at one time. The combination is called a display page. Up to 50 display pages can be entered into the data base at one time. To define these pages, press "D", "D)ISPLAY FORMATS", from the main master menu. The display definition page shown in figure 76 will be presented with 7 options.

To add a display page definition, press "A" then 'ENTER', figure 77 will be displayed and the cursor will be placed under the DW# for data word prompt near the top left of the page. To add the first parameter, just type in its data word number and 'ENTER'. The program will look up the data word and present the parameter number, name, and default display type. Pressing any key except the 'ENTER' key will

toggle between the different allowable display types. Display types can be of five types: DECimal, HEXadecimal, OCTal, BINary, or SWItch. Pressing the 'ENTER' will assign the display type shown and step to the next line. The first four types are self explanatory but the SWItch type is a new feature. SWItch simply tells the user if the data is non-zero, or zero. If it is zero, the "X)ECUTE" display will show the data as "OFF". If it is non-zero, it will show "ON". Designed to be used as event markers for discrete actions such whether a system is turned on or is receiving a signal etc.

There are a few special features which make the display more useful when presented using the "X)ECUTE DISPLAY PAGE" function from main master menu. If two or more entries of the same parameter are entered on sequential lines, the data will be displayed during "X)ECUTE" as follows. If the data types are the same as well as the data word number, sequential frame values will be displayed. For example, to see 5 sequential frame values of data word 1, enter 5 lines of data word number 1 as shown in figure 77 positions 1 thru 5. If the data types are different, then the displayed values will be of the same frame but in the display types indicated. Lines 6 and 7 will show the same frame of "AIRSPED" in both decimal and hexadecimal. The decimal value will be scaled according to the minimum and maximum range of the parameter entered during the parameter setup procedure. The hexadecimal value will be the actual value of the data without scaling. In this way,k the user can check transducer calibrations in any format desired.

When finished describing the display page, press the "CLEAR" key

```
**********************************
*   DISPLAY DEFINITION PAGE      *
**********************************

A)DD NEW DISPLAY PAGE
E)DIT DISPLAY PAGE
I)NSERT DISPLAY PAGE
D)ELETE DISPLAY PAGE
L)IST CURRENT DISPLAY PAGES
P)ARAMETER LISTING
Q)UIT

WHICH ONE->?
```

Figure 75. DSETUP MASTER MENU

```
          ADDING DISPLAY PAGE # 2
**********************************
  POS   DW#   PAR    NAME     DISP
   1     1     4     ALTITUD  DEC
   2     1     4     ALTITUD  DEC
   3     1     4     ALTITUD  DEC
   4     1     4     ALTITUD  DEC
   5     1     4     ALTITUD  DEC
   6     2     5     AIRSPED  DEC
   7     2     5     AIRSPED  HEX
   8     3     6     CAB ALT  DEC
   9     4     7     CAB TEM  DEC
  10    10    25     OAT      DEC
**********************************
 ENTER=NEXT CLEAR=ACCEPT SP=TOG
```

Figure 76. ADD/EDIT DISPLAY PAGE

and the same exit menu is displayed as in the parameter definition procedure.

To edit a previously entered display page, select the "E)DIT DISPLAY PAGE" option from the master menu by pressing "E", then 'ENTER'. This option is handled exactly like the "E)DIT" option from the parameter setup procedure before except that choosing the "L)IST" option will list the display page data base instead of the data word data base. See figures 78 and 79 below.

Both "I)NSERT DISPLAY PAGE" and "D)ELETE DISPLAY PAGE" functions perform exactly the same way as for data word insertion and delete functions explained earlier.

Selecting "L)IST DISPLAY PAGES" from the master menu will bring up the display page listing page shown in figure 78. The display shows a listing of 12 pages which were entered as an example. Because of the Color Computer display is only 32 columns, the listing is rather busy. Along the top, the listing shows how many pages are defined in the system. In this case there are 12 pages, the first ten are shown in the listing. If the "ENTER" key is pressed, the last two pages will be presented as shown in figure 79. The line between the asterisks shows the position number of the ten entries per page. The left column shows the page number with the data word numbers on that page across the line. This format allows the user to see what data words will be displayed when a particular page is displayed in EXECUTE. The user can bring a page into edit by simply typing its number so long as it is shown on the catalog page. Pressing "ENTER" will bring up the next ten pages, if there are ten more. This cycling will continue at the users

```
  CATALOG LISTING OF 12 PAGES
**********************************
PG!1!!2!!3!!4!!5!!6!!7!!8!!9!!10
**********************************
1 1   1   1   1   1   2   2   3   4   10
2 11 12 13 14 15 16 17 18 19 20
3 21 22 23 24 25 26 27 28 29 30
4 31 31 31 31 31 31 31 31 31 31
5 32 32 32 32 32 33 33 33 33 33
6 34 35 36 37 38 39 40 41 42 43
7 11 15 17 19 23 35 43 44 45 46
8 9   9   9   9   9   9   9   9   9   9
9 50 51 52 53 54 55 56 57 58 59
1060 61 62 63 64 65 66 67 68 69
**********************************
NUMBER=SEL, ENTER=CONT, E=EXIT ?
```

Figure 77. DISPLAY PAGE LISTING

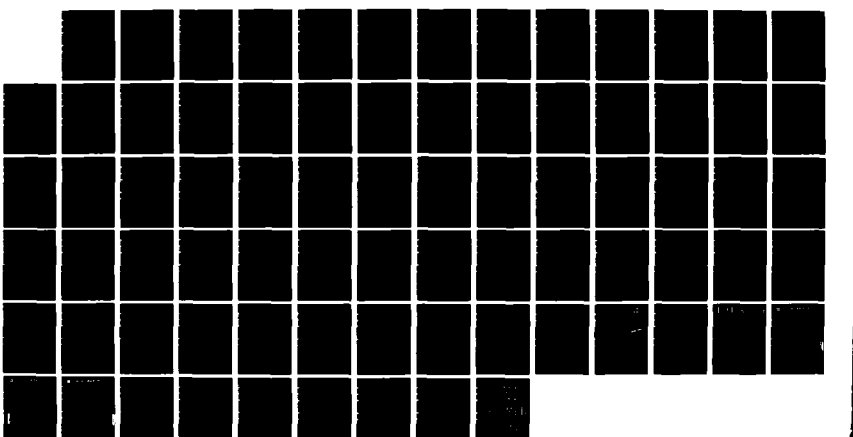
- -

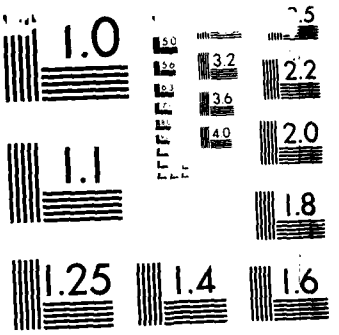
```
  CATALOG LISTING OF 12 PAGES
**********************************
PG!1!!2!!3!!4!!5!!6!!7!!8!!9!!10
**********************************
111   1   1   1   1   2   2   3   4   10
1211 12 13 14 15 16 17 18 19 20
```

```
**********************************
NUMBER=SEL, ENTER=CONT, E=EXIT ?
```

Figure 78. DISPLAY PAGE LISTING cont

AD-A164 835 REAL-TIME FLIGHT TEST PCM DATA ACQUISITION MONITOR(U) 3/3
AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL
OF ENGINEERING J R CROASDALE SEP 85 AFIT/GE/ENG/85S-1
UNCLASSIFIED F/G 17/2 NL





prompt until the entire catalog is displayed. The user can elect to quit using the "EXIT" option. When entered from the "EDIT" option on the master menu page, the catalog will return back to the "EDIT" option if exited.

The "P)ARAMETER LISTING" option can be used to peruse the parameter data base if desired to display previously entered parameter data. This option is exactly the same as the "L)IST DATA WORDS" except that data cannot be recalled into the system. It is used for informational purposes only.

To terminate entering display page definitions, press "Q" from the master menu and the MAIN master menu will appear.

Once all desired parameter and display information is programmed into the data base, the final step is the "X)ECUTE TO BEGIN PCM MONITOR" option. Press "X", then 'ENTER' to select this option.

Displaying The Data

This option is the workhorse of the system. It is responsible for displaying the data set up in previous procedures and presenting it in user readable formats. It also performs the function of providing system status feedback to the user. Calling this option will first prompt the user to set the order switch to either MSB or LSB depending on what was input to the system during the frame specification setup procedure. After a few seconds, the execute master menu will be displayed. This menu is shown in figure 80. Be sure that the PCM test system or encoder is generating the type of PCM set up in the frame specification procedure.

```
*********************************
*      PCM DISPLAY MODULE       *
*********************************

S)TATUS PAGE
B)EGIN DISPLAY PROCESSING
Q)UIT

WHICH ONE->?
```

Figure 79. EXECUTE MASTER MENU

```
*********************************
*   STATUS PAGE, KEY TO EXIT    *
*********************************
                  USER  CALCULATED
BITS/WORD          10       -----
WORDS/FRAME        30       -----
PCM TYPE          AUTO       NRZ
BIT RATE          25.5       25.9
PCM ORDER         MSB       -----
PCM POLARITY      NORMAL    -----
SYNC ERRORS       -----      10
PCM HEALTH        -----      3
SYNC WORDS 1- 1111101011
   INVERSE 2- 1100110011
IF IN SYNC 3- 0100000000
```

Figure 80. STATUS DISPLAY PAGE

Calling the "S)TATUS" option brings up the status display page shown in figure 81. This page is for display only, no input is possible. Pressing any key returns the user to the master menu. Much of the display is the frame specification entered earlier. The two columns, "USER" and "CALCULATED" display the data entered by the user, and calculated by the system respectively. Where items are not calculated, the display is filled with dashes. If the sample data was entered correctly during the frame setup procedure, it should be the same as that in figure 81. The four types of information returned by the PCM processor are shown in the "CALCULATED" column. These values will be returned whether the user selects any of the automatic features or not. In the sample, 25.5Khz was selected as the manual bit rate and the system calculated 25.9Khz. If there is too much difference between these two rates in the actual system, check the frequency of the PCM encoder or use Auto calculation mode. The main reason for allowing manual entry of the PCM signal rate is to determine if the PCM signal is erratic or stable. If the system has trouble locking on to the signal, this may be an indication of why.

"PCM ORDER" and "POLARITY" are user set and not calculated. "SYNC ERRORS" are generated once sync is established and then broken. If no signal is present, then no sync errors will occur since the signal was never synchronized. If however, a good signal was obtained, the error counter in the system is set and incremented each time a break lock is encountered. The only way to reset it is to exit the EXECUTE program and re-enter.

"PCM HEALTH" is a generic term which tells the user how stable the

signal is. This calculation is explained in the PCM Processor software section but results in a number from Ø to 1Ø with Ø being a good stable signal and 1Ø being rather erratic. An absolute value cannot be obtained from this except that it, along with the sync error counter above, tells the user if the signal is erratic or stable. The health calculation is actually the number of counts that have to be added to the minimum count to keep the signal synchronized.

The last three lines on the status page are for the sync words. They are repeated here for user information. The word "SYNC" in "SYNC WORDS" will be written in inverse video if the PCM stream is synchronized. If it isn't, it will be displayed in normal video. The difference between inverse and normal depends on the type of display

```
   ENGINEERING DISPLAY PAGE 1
**********************************
DW#   NAME     VALUE      UNITS TY
**********************************
1     ALTITUD  13429.41   FEET   D
1     ALTITUD  13433.23   FEET   D <-
1     ALTITUD  13440.44   FEET   D
1     ALTITUD  13445.94   FEET   D
1     ALTITUD  13452.22   FEET   D
2     AIRSPED  250.4      KNOTS  D
2     AIRSPED  ØØFE       HEX    H
3     CAB ALT  8Ø18       FEET   D
4     CAB TEM  78         DEG F  D
1Ø    OAT      -22.5      DEG C  D
**********************************
<> NUM, T)YP, B)AR, P)LT, E)XIT
```

Figure 81. ENGINEERING DISPLAY PAGE

being used. Most video monitors display white characters on a black background so inverse is black characters on a white background. If a television set is used, the display is black on a white background so inverse is white on black. In all cases, the word "SYNC" will be displayed opposite from all other words on the screen if the signal is in sync.

After returning the master menu, the only other option remaining except quitting, is to display the data. Selecting the "B)EGIN DISPLAY PROCESSING" option brings up the engineering display page shown in figure 82.

When the engineering data display page appears, you are asked to input a page number. Select "1" then 'ENTER' and the system will bring in the display page 1 information entered earlier. The sample should show the page defined earlier. At this point, the data is being displayed and continually updated as fast as the display processor program can run. The data is updated from top to bottom on-the-fly. The format type for each data word is shown in the far right column.

When the value is displayed in decimal, the data is scaled and offset to engineering data reflecting the minimum and maximum range described during the parameter setup procedure. At any other setting, such as HEX for example, the data is presented in raw data as it is read in from the PCM processor. The units column shows the selected conversion type if other then decimal to alert the user that the data is not presented in engineering units.

To change the displayed conversion type, press the "T)yp" function shown at the bottom of the page. The data types will toggle for the

selected line. The selected line is shown by the inverse character in the far right column under the heading TY for TYPE. The example shows this as an arrow because typewriters can't type in inverse. The type will change from "D" to "H", "O", "B", and "S", and then revert back to "D". All the time, the program continues to update the data values in whatever type is selected. To change the selected line, use the arrow keys to scroll the inverse select symbol.

Pressing the right or left arrow brings in the next or previous page for display. Pressing a number puts the user in the page number entry field at the top of the page so he/she can manually select a page for display. During the process of bringing in new pages, data update is temporarily suspended but begins automatically when the page is displayed again.

Two other very important functions of the EXECUTE program are the bar graph and "P)LT" (plot) functions. Selecting the "B)AR" function displays the page in bar graph format. See Figure 83. Although there is no user interaction in the bar graph display page some of the features are described.

The top line of the display shows which page is being displayed. In this example it is display page 1 consisting of the same parameters shown on engineering page 1. In fact, whatever page is shown on the engineering display page will be shown on the bar graph page. The only way to get to the bar graph is through the engineering page. This holds true for the plot page shown next. The vertical bars are the alarm limits in percent of full scale which were set up during the parameter setup procedure. AT the left side of each bar is a mark

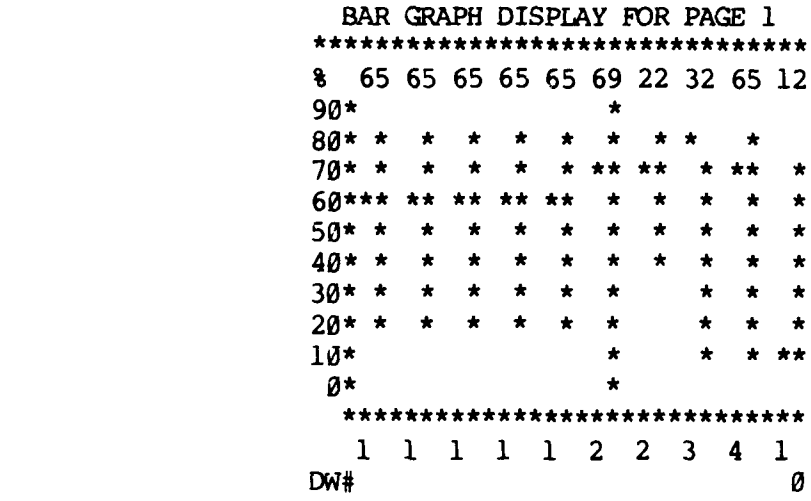


Figure 82. BAR GRAPH DISPLAY

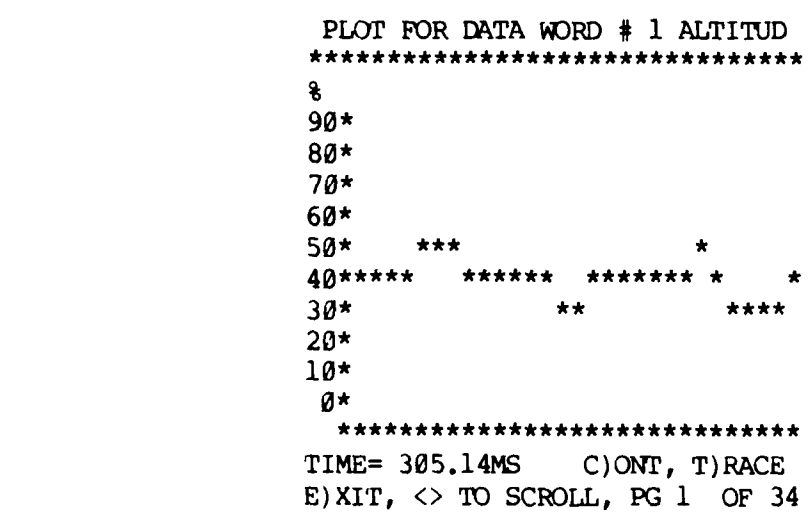


Figure 83. PLOT DISPLAY PAGE

which shows a rough value in percent of the data word number shown at the bottom of the page. Notice that the value for data word #3 is above the alarm limits. In this case, the user can elect to monitor only that data word using the P)LOT function described below. Above the bar at the top of the page is the actual percent value of the data word.

After any key is pressed, the user is returned to the engineering display page. From here, he/she can elect to see a plot of the selected value. The selected value is shown in inverse at the right hand column of the page, the TY column. Selecting "P)LT" puts the user on the page shown in figure 84. This page is similar to the bar graph page however only one data word is plotted at a time. Each point represents one frame. The intent of this feature is to look at a particular data word in detail for intermittent spikes etc. The sample rate, or time between samples as it is used here, is a function of the data rate and words per frame and is shown at the bottom of the page in microseconds.

There are two user selected modes of operation in plot: "C)ONT" for continuous, and "T)RACE". In "C)ONT" mode, the plot routine scans each frame in the data buffer for the particular data word being plotted. It keeps updating the value in a continual basis until halted by the user. In this way, the plot is kept updated with the latest data input to the system. Because the display processor is slow compared to the PCM processor due to the BASIC language, a particular frame could be updated before the display processor displays it and data could be lost. To solve this problem, the "T)RACE" feature was

added.

"T)RACE" commands the PCM processor to fill the memory buffer with one and only one data word. When the buffer is full, it stops and waits for the user to begin another trace. Up to 1000 data words can be stored in the memory buffer at one time so up to 1000 frames will be captured. Each frame will be continuous with no data loss. The user can scan the buffer 29 frames at a time by using the left and right arrows. This equates to 1000/29 or 34 pages of information. By pressing "C)ONT", the user will be returned to the "C)ONT" function. By pressing "T)RACE", a new trace will begin. Pressing "E)XIT" returns the user to the engineering display. While scrolling through the 34 pages of trace data. If a spike is present on the plot, its time can be roughly measured by counting the sample times. This will have to be done manually at this time but an automatic time measurement system using cursors could added later. There is a problem and that is if the data rate is very slow and the number of words per frame is very large, the time to fill the buffer will be quite long. For example, if the words were 16 bits wide, the number of words per frame were 200, and the bit rate was 1Khz, it would take 3.2 seconds per frame which equates to 53.33 minutes. For this reason, the "T)RACE" algorithm will display a page when it is ready and not wait until the memory is full. In fact, the user can see the data being plotted as it is read in if he/she keeps scrolling fast enough. A more reasonable set of data such as 10 bits per word, and 30 words per frame at 25 Khz would yield one frame per .3 sec or 5 minutes to fill memory. This is not a limitation of the PCM monitor system, only the data rates and PCM specifications.

Manual Reset

The manual reset function is performed by depressing the reset button on the PCM processor card. This function should be used if for some reason the system hangs or for test purposes. Since the PCM processor card program is stored in ROM but executed in RAM, a memory fault may cause the system to malfunction. Manually resetting the system causes a reload of the RAM. There is no other practical reason for resetting the system manually. The display processor must be re-executed once the PCM processor has been reset.

Exiting The System

To exit the PCM monitor there are several methods. The best method is to "EXIT" the program running and then "QUIT" from the MAIN master menu. A quicker way is to open the disk drive door, remove the disk and shut off the Color Computer. Since all data is stored on disk, power failures or accidentally pressing the 'BREAK' or resetting the computer will not harm it unless, of course, something happens during the time when the system is storing data to disk. If that happens, some data may be lost but most should still be intact.

References Used in This Manual

1. Croasdale, John R. Real-Time Flight Test PCM Acquisition Monitor. AFIT Thesis 85S-1.

2. Getting Started With Extended color Basic, Tandy corporation, FortWorth, Texas, 1984.

3. Radio Shack TRS80 Color Computer Disk System Owner's Manual and Programming Guide, Tandy corporation, FortWorth, Texas, 1981.

Appendix C

REAL-TIME FLIGHT TEST PCM

DATA ACQUISITION MONITOR

PROGRAM LISTINGS

BY

JOHN R. CROASDALE, LTC, USAF

```
LOCATION OBJECT CODE LINE     SOURCE LINE

                              1                "68000"
                              2                ORG 0000H
                              3 ********************************************************************
                              4 *                                                                  *
                              5 *                      PCM MONITOR PROGRAM                         *
                              6 *                      VERSION 1.0, SEPT 85                        *
                              7 *  THIS PROGRAM IS INSTALLED INTO READ ONLY MEMORY (ROM)           *
                              8 *  LOCATED INT HE PCM MONITOR CARD.  ITS PRIME FUNCTION IS         *
                              9 *  TO CONTROL THE PCM MONITOR CIRCUITRY AND TRANSFER THE PCM       *
                             10 *  DATA FROM THE INPUT REGISTER TO THE MEMORY BUFFER START-        *
                             11 *  ING AT $C000.  THIS BUFFER IS 1K WORDS IN LENGTH BUT THE        *
                             12 *  MC68000 ADDRESSES BYTES ONLY, THEREFORE THE ADDRESS RANGE       *
                             13 *  OF THIS PROGRAM IS FROM  $C000 TO $C7FF (2K BYTES).             *
                             14 *  THE PCM MONITOR IS DESIGNED TO ACCESS WORDS ONLY THERE-         *
                             15 *  FOR IT IS IMPERATIVE NOT TO TRY TO READ OR WRITE TO ODD         *
                             16 *  ADDRESSES.  UNPREDICTABLE RESULTS MAY OCCUR.                    *
                             17 *                                                                  *
                             18 *  BECAUSE OF THE SPEED REQUIRED BY THE SYSTEM, MAXIMUM CON-       *
                             19 *  SIDERATION IS GIVEN TO THE TIME UTILIZED BY EACH INSTRUC-       *
                             20 *  TION.  MOST OF THE TIME CRITICAL FUNCTIONS THEREFORE USE        *
                             21 *  REGISTERS, D0-D7, AND A0-A3, AS MUCH AS POSSIBLE.  IN           *
                             22 *  ADDITION, THE ZERO PAGE IS USED TO AVOID UNNECESSARY            *
                             23 *  MEMORY ACCESSES.  ZERO PAGE IS FROM $0000-$7FFF AND FROM        *
                             24 *  $FFFF8000-$FFFFFFFF.                                            *
                             25 *                                                                  *
                             26 ********************************************************************
                             27
                             28 *  EQUATES
                             29
<FFFFC7FE>                   30 VALID          EQU 0FFFFC7FEH   COMMAND FLAG TO PROCESSOR
<FFFFC7FC>                   31 USERATE        EQU 0FFFFC7FCH   USER INPUT PCM RATE, 0=CAL
<FFFFC7FA>                   32 PCMTYPE        EQU 0FFFFC7FAH   USER INPUT PCM TYPE, "C"=CAL
<FFFFC7F8>                   33 BITPWRD        EQU 0FFFFC7F8H   USER INPUT FOR LENGTH OF WORD
<FFFFC7F6>                   34 FRAMLEN        EQU 0FFFFC7F6H   USER INPUT FOR WORDS/FRAME
<FFFFC7F4>                   35 FRASYN3        EQU 0FFFFC7F4H   LAST FRAME SYNC WORD
<FFFFC7F2>                   36 FRASYN2        EQU 0FFFFC7F2H   SECOND FRAME SYNC WORD
<FFFFC7F0>                   37 FRASYN1        EQU 0FFFFC7F0H   FIRST FRAME SYNC WORD
<FFFFC7EE>                   38 ERRCNT         EQU 0FFFFC7EEH   ERROR ACCUMULATOR
<FFFFC7EC>                   39 PLARITY        EQU 0FFFFC7ECH   USER INPUT FOR POLARITY "N"=NORM
<FFFFC7EA>                   40 FRPBUFF        EQU 0FFFFC7EAH   USER INPUT FOR FRAMES PER BUFFER
<FFFFC7E8>                   41 CALTYPE        EQU 0FFFFC7E8H   RETURNED CALCULATED PCM TYPE
<FFFFC7E6>                   42 CALRATE        EQU 0FFFFC7E6H   RETURNED CALCULATED PCM RATE
<FFFFC7E4>                   43 CORRECT        EQU 0FFFFC7E4H   RETURNED VALUE OF PCM CORRECTION
<FFFFC7E2>                   44 STATUS         EQU 0FFFFC7E2H   RETURNED STATUS, 0=NOT IN SYNC
<FFFFC7E0>                   45 DATADD         EQU 0FFFFC7E0H   ADD POINTER TO BUFFER
                             46 *
                             47 *
                             48 * SYSTEM FUNCTION ADDRESSES
                             49 *
```

```
                <2000>     50 RATECNT          EQU 2000H          BIT TIMER INPUT, READ ONLY
                <4000>     51 HLFCNTR          EQU 4000H          HALF COUNTER OUTPUT
                <6000>     52 ONECNTR          EQU 6000H          ONE COUNTER OUTPUT
            <FFFF8000>     53 SETTYPE          EQU 0FFFF8000H     WRITE SETS PCM, READ AT BIT RATE
            <FFFFA000>     54 WORDLEN          EQU 0FFFFA000H     WRITE ST WD CNTR, RD @ WD RATE
            <FFFFC000>     55 BUFFER           EQU 0FFFFC000H     START OF BUFFER ADDRESS
                           56 *
                           58 *  OTHER EQUATES
                           59 *
            <FFFFF800>     60 RSTART           EQU 0FFFFF800H     START OF RAM PROGRAM ADDRESS
                           61 *
                           62 *
                           63 * START OF PROGRAM
                           64 *
                           65 * FIRST SET STACK AND PROGRAM COUNTER
                           66 *
000000 FFFF                67                  DC.W 0FFFFH        SET STACK TO TOP OF RAM
000002 FFFE                68                  DC.W 0FFFEH
000004 0000                69                  DC.W 0000          SET PROGRAM COUNTER
000006 0008                70                  DC.W 0008
                           71 * NEXT MOVE ROM IMAGE OF PROGRAM INTO RAM
000008 307C 0018           72 BEGIN            MOVE.W #START,A0
00000C 327C F800           73                  MOVE #RSTART,A1
000010 32D8                74 LOOP1            MOVE [A0]+,[A1]+
000012 B0FC 1000           75                  CMP  #1000H,A0     MOVE 2048 BYTES
000016 65F8                76                  BCS.S LOOP1
000018 31FC FFFF           77 START            MOVE #-1,VALID     FIRST INITIALIZE COUNTERS ETC
00001E 4278 C7E4           78                  CLR  CORRECT
000022 4278 C7EE           79                  CLR  ERRCNT
000026 4278 C7E2           80 USRECAL          CLR  STATUS
00002A 4278 C7E6           81                  CLR  CALRATE
00002E 4A78 C7FE           82 WAIT1            TST  VALID         WAIT FOR USER TO START
000032 6BFA                83                  BMI.S WAIT1         NOT READY YET SO WAIT
000034 4A78 C7FE           84 WAIT2            TST  VALID         USER RECALL ENTRY
000038 6BEC                85                  BMI.S USRECAL
00003A 6176                86 DORATE           BSR.S RATE          CALCULATE PCM RATE
00003C 0C40 FF00           87                  CMP  #0FF00H,D0     IS THERE A SIGNAL?
000040 64F2                88                  BCC.S WAIT2        NO, THEN KEEP TRYING
000042 31C0 C7E6           89                  MOVE D0,CALRATE     TELL USER WHAT THE RATE IS
000046 4A78 C7FC           90                  TST  USERATE       USER INPUT RATE?
00004A 6704                91                  BEQ.S CONT1         NO, THEN USE OURS
00004C 3038 C7FC           92                  MOVE USERATE,D0     YES, THEN USE HIS
000050 D078 C7E4           93 CONT1            ADD  CORRECT,D0     ADD CORRECTION IF ANY
000054 31C0 6000           94                  MOVE D0,ONECNTR     SET ONE COUNTER
000058 E248                95                  LSR  #1,D0         DIVIDE BY 2
00005A 31C0 4000           96                  MOVE D0,HLFCNTR     AND SET HALF COUNTER
00005E 616A                97                  BSR.S TYPE          CALCULATE PCM TYPE
000060 31C1 C7E8           98                  MOVE D1,CALTYPE     TELL USER WHAT WE FOUND
000064 0C78 0042           99                  CMP  #"B",PCMTYPE A USER FORCED TYPE?
00006A 6704               100                  BEQ.S CONT2         NO, THEN USE OURS
00006C 3238 C7FA          101                  MOVE PCMTYPE,D1     YES, USE HIS
000070 0A41 FFFF          102 CONT2            EOR  #-1,D1        TRICK TO MAKE TYPE EVEN OR ODD
000074 E449               103                  LSR  #2,D1
000076 31C1 8000          104                  MOVE D1,SETTYPE     SET THE TYPE
```

```
00007A 3838 C7F6    105                 MOVE FRAMLEN,D4   SET UP REGISTERS FOR READ ROUTINE
00007E 3A38 C7F4    106                 MOVE FRASYN3,D5
000082 3C38 C7F2    107                 MOVE FRASYN2,D6
000086 3E38 C7F0    108                 MOVE FRASYN1,D7
00008A 307C 8000    109                 MOVE #SETTYPE,A0  THIS ADDRESS READS AT BIT RATE
00008E 327C A000    110                 MOVE #WORDLEN,A1  THIS ADDRESS READS AT WRD RATE
000092 6164         111                 BSR.S READ         BEGIN READING DATA
000094 4A78 C7FE    112                 TST  VALID        IS IT A USER RECALL?
000098 6B00 FF6E    113                 BMI  BEGIN        YES THEN START OVER
00009C 5278 C7EE    114                 ADDQ #1,ERRCNT    INCREMENT ERROR COUNTER
0000A0 5278 C7E4    115                 ADDQ #1,CORRECT   ALSO CORRECTION
0000A4 0C78 000A    116                 CMP  #10,CORRECT  MAX CORRECTION IS 9
0000AA 658E         117                 BCS.S DORATE       TRY AGAIN
0000AC 4278 C7E4    118                 CLR  CORRECT
0000B0 6088         119                 BRA.S DORATE
                    120 *
                    121 *
                    122 *  RATE ROUTINE CALCULATES PCM RATE BY GETTING  MINIMUM COUNT
                    123 *  FROM BIT TIMER
                    124 *
0000B2 303C FFFF    125 RATE            MOVE #-1,D0       SET ACCUMULATOR TO MAX
0000B6 3400         126                 MOVE D0,D2        GET MINIMUM COUNT IN D0
0000B8 3238 2000    127 LOOPR           MOVE RATECNT,D1
0000BC B240         128                 CMP  D0,D1        IS D0 MINIMUM?
0000BE 6402         129                 BCC.S CONTR1       NO THEN KEEP TRYING!
0000C0 3001         130                 MOVE D1,D0        YES, THEN MAKE IT SO
0000C2 5342         131 CONTR1          SUBQ #1,D2        DECREMENT COUNTER
0000C4 66F2         132                 BNE.S LOOPR        KEEP LOOKING IF NOT DONE
0000C6 E248         133                 LSR  #1,D0        CONVERT TO BIT TIME
0000C8 4E75         134                 RTS               AND RETURN
                    135 *
                    136 *
                    137 *  TYPE ROUTINE CALCULATES PCM TYPE BY COMPARING RATE WITH
                    138 *  TWICE THE MAX COUNT.  IF LESS THEN MUST BE BI-PHASE
                    139 *
0000CA 4278 8000    140 TYPE            CLR  SETTYPE      SET NRZ MODE
0000CE 6112         141                 BSR.S  MAXCNT       GET MAXIUM COUNT
0000D0 323C 0042    142                 MOVE #"B",D1      PRIME PUMP FOR BI-PHASE
0000D4 E448         143                 LSR  #2,D0        GET MAX BIT TIME
                    144 *                                                          *
0000D6 B078 C7E6    145                 CMP  CALRATE,D0    IS IT LESS THAN HALF?
0000DA 6504         146                 BCS.S CONTT        YES, THEN BI_PHASE
0000DC 323C 004E    147                 MOVE #"N",D1      NO, THEN SELECT NRZ
0000E0 4E75         148 CONTT           RTS
0000E2 4240         149 MAXCNT          CLR  D0           SET ACCUMULATOR TO MIN
0000E4 343C FFFF    150                 MOVE #-1,D2
0000E8 3238 2000    151 LOOPM           MOVE RATECNT,D1    READ RATE
0000EC B240         152                 CMP  D0,D1        GET MAX COUNT IN D0
0000EE 6502         153                 BCS.S CONTM
0000F0 3001         154                 MOVE D1,D0
0000F2 5342         155 CONTM           SUBQ #1,D2
0000F4 66F2         156                 BNE.S LOOPM        NOT DONE, THEN CONTINUE
0000F6 4E75         157                 RTS
0000F8 4240         158 READ            CLR  D0           CALCULATE MASK, PUT IN D0
```

```
0000FA 3238 C7EA    159                 MOVE FRPBUFF,D1   DO SO BY SHIFTING X INTO D0
0000FE 3438 C7F8    160                 MOVE  BITPWRD,D2  USE D2 AS COUNTER
000102 007C 0010    161 LOOPMA          OR   #16,SR       SET X FLAG IN SR
000106 E350         162                 ROXL #1,D0        SHIFT IT IN
000108 5342         163                 SUBQ #1,D2        ARE ALL BITS SHIFTED IN
00010A 6AF6         164                 BPL.S LOOPMA
                    165 *
                    166 *
                    167 *  SYNCHRONIZE THE BIT STREAM
                    168 *
00010C 343C FFFF    169                 MOVE #-1,D2       USE D2 AS A MAX TRY COUNTER
000110 347C C000    170 SYNC            MOVE #BUFFER,A2   A2 IS BUFFER POINTER
000114 3610         171 SYNC1           MOVE [A0],D3      READ AT BIT RATE
000116 C640         172                 AND  D0,D3        MASK UNWANTED BITS
000118 B647         173                 CMP  D7,D3        D7 HAS FIRST SYNC WORD
00011A 57CA FFF8    174                 DBEQ D2,SYNC1     CONTINUE FOR 65535 BITS
00011E 6706         175                 BEQ.S CONTS1       A MATCH, THEN LOOK FOR MORE
000120 4278 C7E2    176 RTSERR          CLR  STATUS       NO MATCH, NO SYNC
000124 4E75         177                 RTS               RETURN AS AN ERROR
000126 32B8 C7F8    178 CONTS1          MOVE BITPWRD,[A1]  START WORD COUNTER
00012A 4A46         179 SYNC2           TST  D6           IS THERE A SECOND SYNC?
00012C 6714         180                 BEQ.S READATA      NO SO BEGIN READING DATA
00012E 3611         181                 MOVE [A1],D3      YES SO READ IT AT WORD RATE
000130 C640         182                 AND  D0,D3        MASK IT
000132 B646         183                 CMP  D6,D3        IS IT THERE?
000134 66EA         184                 BNE.S RTSERR       NO, THEN ERROR
000136 4A45         185 SYNC3           TST  D5           IS THERE A THIRD SYNC?
000138 6708         186                 BEQ.S READATA      NO, SO READ DATA
00013A 3611         187                 MOVE [A1],D3      THIRD SYNC?
00013C C640         188                 AND  D0,D3
00013E B645         189                 CMP  D5,D3
000140 66DE         190                 BNE.S RTSERR       NO, THEN ERROR
000142 31FC FFFF    191 READATA         MOVE #-1,STATUS   TELL USER WE'RE IN SYNC
000148 4A78 C7FE    192                 TST  VALID        IS USER RECALLING US?
00014C 6BD2         193                 BMI.S RTSERR       YES, THEN RETURN
00014E 662E         194                 BNE.S TRACE        IF NOT 0 THEN MUST BE TRACE
000150 3838 C7F6    195 CONTOUS         MOVE FRAMLEN,D4    USE D4 AS WORD COUNTER
000154 3491         196 LOOPR1          MOVE [A1],[A2]  READ AND STORE RAW DATA
000156 0C78 004E    197                 CMP  #"N",PLARITY   NORMAL POLARITY?
00015C 6704         198                 BEQ.S CONTRD1      YES SO DON'T INVERT
00015E 0A52 FFFF    199                 EOR  #-1,[A2]     NO, THEN MUST BE INVERSE
000162 C15A         200 CONTRD1         AND  D0,[A2]+     MASK DATA AND INCREMENT PNTR
000164 5344         201                 SUBQ #1,D4        WORD=WORD-1
000166 66EC         202                 BNE.S LOOPR1       FRAME NOT DONE SO CONTINUE
000168 3611         203                 MOVE [A1],D3      FRAME DONE SO LOOK FOR SYNC1
00016A C640         204                 AND  D0,D3        MASK OF COURSE]
00016C B647         205                 CMP  D7,D3        GOOD SYNC?
00016E 66B0         206                 BNE.S RTSERR       NO, SO RETURN WITH ERROR
000170 5341         207                 SUBQ #1,D1        IS THIS THE LAST FRAME?
000172 66B6         208                 BNE.S SYNC2        NO SO CONTINUE SYNC SEARCH
000174 347C C000    209                 MOVE #BUFFER,A2    YES, THEN RESET POINTERS]
000178 3238 C7EA    210                 MOVE FRPBUFF,D1
00017C 60AC         211                 BRA.S SYNC2       AND CONTINUE SYNC SEARCH
                    212 *
```

```
                        213 *
                        214 *  TRACE MODE FILLS BUFFER WITH ONE DATA WORD, THE ONE STORED
                        215 *  IN THE VALID FLAG
                        216 *
00017E 383C 0001        217 TRACE           MOVE #1,D4      USE D4 AS WORD POINTER
000182 3611             218 LOOPT1          MOVE [A1],D3    READ DATA WORD
000184 B878 C7FE        219                 CMP  VALID,D4   ARE WE POINTING TO RIGHT WORD?
000188 6704             220                 BEQ.S CONTT1     YES, TO STORE IT
00018A 5244             221                 ADDQ #1,D4      INCREMENT POINTER AND
00018C 60F4             222                 BRA.S LOOPT1     KEEP LOOKING
00018E 3483             223 CONTT1          MOVE D3,[A2]    STORE DATA WORD
000190 0C78 004E        224                 CMP  #"N",PLARITY  CHECK POLARITY
000196 6704             225                 BEQ.S CONTT2     NORMAL
000198 0A52 FFFF        226                 EOR  #-1,[A2]   INVERSE IT IF NECESSARY
00019C 31CA C7E0        227 CONTT2          MOVE A2,DATADD  TELL USER WHERE WE'RE AT
0001A0 C15A             228                 AND  D0,[A2]+   MASK AND INCREMENT
0001A2 B4F8 C7D0        229                 CMP  BUFFER+07D0H,A2  IS BUFFER FULL
0001A6 641A             230                 BCC.S WAIT3      YES, SO WAIT FOR USER
0001A8 343C FFFF        231                 MOVE #-1,D2     RESET TRY COUNTER
0001AC B878 C7F6        232                 CMP  FRAMLEN,D4  IF DATA WORD IS NOT THE
0001B0 6600 FF62        233                 BNE  SYNC1      LAST THEN MUST RE-SYNC
0001B4 3611             234                 MOVE [A1],D3    IF IT IS, THEN NOT
0001B6 C640             235                 AND  D0,D3      NECESSARY
0001B8 B647             236                 CMP  D7,D3      IS FIRST SYNC NEXT
0001BA 6600 FF64        237                 BNE  RTSERR     NO, THEN AN ERROR
0001BE 6000 FF6A        238                 BRA  SYNC2      YES, THEN CONTINUE SYNC SEARCH
                        239 *
                        240 *
                        241 *  WAIT LOOP FOR USER TO START ANOTHER TRACE OR STOP
                        242 *
0001C2 31FC FFFF        243 WAIT3           MOVE #-1,VALID  TELL USER WE'RE WAITING
0001C8 4A78 C7FE        244 LOOPW1          TST  VALID
0001CC 6BFA             245                 BMI.S LOOPW1
0001CE 31FC C000        246                 MOVE #BUFFER,DATADD  READY TO TRY AGAIN
0001D4 6000 FF3A        247                 BRA  SYNC            SO RESET AND RE-SYNC
                        248                 END
```

```
                        00100 ************************************************
                        00110 *                                              *
                        00120 *                    CONVERT                   *
                        00130 *                                              *
                        00140 * THIS PROGRAM DOES THREE FUNCTIONS.           *
                        00150 * AFTER READING A NUMBER FROM "INPUT" AND:     *
                        00160 * IF D=1 THEN IT DISPLAYS THE NUMBER IN HEX    *
                        00170 * IF D=2 THEN IT DISPLAYS THE NUMBER IN OCTAL  *
                        00180 * IF D=3 THEN IT DISPLAYS THE NUMBER IN BINARY *
                        00190 * IF D<>1, 2, OR 3 THEN IT JUST RETURNS       *
                        00200 *                                              *
                        00210 * THE NUMBER IS DISPLAYED AT ADDRESS "SCREEN"  *
                        00220 * BOTH "INPUT" AND "SCREEN" ARE PASSED TO THE  *
                        00230 * ROUTINE AT LOCATION $600 - $603 AND IN THE D *
                        00240 * REGISTER USING THE INTCNV ROUTINE FROM BASIC *
                        00250 *                                              *
                        00260 ************************************************
7000                    00270          ORG     $7000    *USR1 IS CALLED FROM 2048
            0600        00280 INPUT    EQU     $0600    *VARIABLE HIGH ADDRESS (VH)
            0602        00290 BPWSTG   EQU     $0602
            0604        00300 TEMP     EQU     $0604
            B3ED        00310 INTCNV   EQU     $B3ED    *CALL HERE RETURNS INT IN D
            A002        00320 COUT1    EQU     $A002    *INDIRECT CALL OUTPUTS CHAR
                        00330                           *AT CURSOR POSITION IN $88
                        00340
7000 BD     B3ED        00350 START    JSR     INTCNV
7003 C1     01          00360          CMPB    #$01
7005 27     09          00370          BEQ     HEXOUT
7007 C1     02          00380          CMPB    #$02
7009 27     24          00390          BEQ     OCTOUT
700B C1     03          00400          CMPB    #$03
700D 27     62          00410          BEQ     BINOUT
700F 39                 00420          RTS             *EXIT IF INVALID CALL
                        00430 ************************************************
                        00440 *                                              *
                        00450 *                HEXIDECIMAL OUT               *
                        00460 *                                              *
                        00470 * THIS PROGRAM CONVERTS A HEXIDECIMAL NUMBER IN *
                        00480 * REGISTER D TO A HEX ASCII OUTPUT             *
                        00490 *                                              *
                        00500 * CORRECT NUMBER OF BITS.  UNUSED BITS ARE SET *
                        00510 ************************************************
7010 EC     9F 0600     00520 HEXOUT   LDD     [INPUT] *GET DATA
7014 8D     05          00530          BSR     PRBYTE  *PRINT A IN ASCII
7016 1F     98          00540          TFR     B,A
7018 8D     01          00550          BSR     PRBYTE
701A 39                 00560          RTS             *EXIT
701B 34     02          00570 PRBYTE   PSHS    A       *GET FIRST BYTE
701D 44                 00580          LSRA
```

```
701E 44          00590          LSRA
701F 44          00600          LSRA
7020 44          00610          LSRA
7021 8D 02       00620          BSR     PRHEX
7023 35 02       00630          PULS    A
7025 84 0F       00640 PRHEX    ANDA    #$0F     *USE LOW NIBBLE ONLY
7027 81 0A       00650          CMPA    #$0A     *NUMBER OR CHAR?
7029 25 6D       00660          BCS     COUT
702B 8B 07       00670          ADDA    #$07     *SKIP UNWANTED CHARACTERS
702D 20 69       00680          BRA     COUT
                 00690
                 00700 ************************************************
                 00710 *                                              *
                 00720 *                 OCTAL OUT                    *
                 00730 *                                              *
                 00740 ************************************************
702F EC 9F 0600  00750 OCTOUT   LDD     [INPUT] *GET DATA
7033 8E 000F     00760          LDX     #$0F     *OUTPUT 16 BITS, 1ST IS FREE
7036 FD 0604     00770          STD     TEMP     *SAVE FOR LATER
7039 4F          00780          CLRA             *ROTATE FIRST BIT INTO A
703A 5F          00790          CLRB             *USE B AS LEADING 0 INDICATOR
703B 78 0605     00800          LSL     TEMP+1
703E 79 0604     00810          ROL     TEMP
7041 49          00820          ROLA
7042 27 04       00830          BEQ     OCONT1   *DON'T OUTPUT LEADING 0'S
7044 CB 01       00840          ADDB    #$01     *MARK B AS NO MORE LEADING 0'S
7046 8D 50       00850          BSR     COUT
7048 4F          00860 OCONT1   CLRA             *GET READY FOR NEXT NIBBLE
7049 8D 10       00870          BSR     ROTATE   *GET 3 BITS IN A
704B 26 05       00880          BNE     OCONT2
704D 5D          00890          TSTB             *IF 0 IS IT LEADING?
704E 26 02       00900          BNE     OCONT2
7050 20 04       00910          BRA     OCONT3
7052 CB 01       00920 OCONT2   ADDB    #$01     *MARK
7054 8D 42       00930          BSR     COUT
7056 30 1D       00940 OCONT3   LEAX    -3,X
7058 26 EE       00950          BNE     OCONT1   *GET NEXT NIBBLE
705A 39          00960          RTS
705B 78 0605     00970 ROTATE   LSL     TEMP+1
705E 79 0604     00980          ROL     TEMP
7061 49          00990          ROLA
7062 78 0605     01000          LSL     TEMP+1
7065 79 0604     01010          ROL     TEMP
7068 49          01020          ROLA
7069 78 0605     01030          LSL     TEMP+1
706C 79 0604     01040          ROL     TEMP
706F 49          01050          ROLA
7070 39          01060          RTS
                 01070
```

```
                        01080 ************************************************
                        01090 *                                              *
                        01100 *                    BINARY OUT                *
                        01110 *                                              *
                        01120 * THIS PROGRAM CONVERTS A HEXIDECIMAL NUMBER IN  *
                        01130 * REGISTER D TO A BINARY ASCII OUTPUT IN OUTBUF  *
                        01140 * IT CONSULTS BITS PER WORD (BPW) AND OUTPUTS A  *
                        01150 * CORRECT NUMBER OF BITS.  UNUSED BITS ARE SET   *
                        01160 * TO SPACES                                     *
                        01170 *                                              *
                        01180 ************************************************
                        01190 *
7071 EC    9F 0600      01200 BINOUT  LDD     [$0600] *READ DATA
7075 8E    0010         01210         LDX     #$10
7078 BC    0602         01220 BLOOP1  CMPX    BPWSTG
707B 27    06           01230         BEQ     BCONT1  *IF X HAS # BITS THEN START
707D 30    1F           01240         LEAX    -1,X
707F 58                 01250         LSLB            *SHIFT OUT UNWANTED BIT
7080 49                 01260         ROLA
7081 20    F5           01270         BRA     BLOOP1
7083 58                 01280 BCONT1  LSLB            *GET BIT
7084 49                 01290         ROLA
7085 34    02           01300         PSHS    A       *SAVE FOR LATER
7087 24    04           01310         BCC     BCONT2
7089 86    01           01320         LDA     #$01    *CARRY SET = A "1"
708B 20    02           01330         BRA     BCONT3
708D 86    00           01340 BCONT2  LDA     #$00
708F 8D    07           01350 BCONT3  BSR     COUT    *OUTPUT CHAR
7091 35    02           01360         PULS    A
7093 30    1F           01370         LEAX    -1,X
7095 26    EC           01380         BNE     BCONT1
7097 39                 01390         RTS             *DONE THEN EXIT
                        01400 ************************************************
                        01410 *                                              *
                        01420 *                 CHARACTER OUT                *
                        01430 *                                              *
                        01440 * THIS ROUTINE DISPLAYS A NUMBER TO THE TEXT    *
                        01450 * SCREEN AT LOCATION PASSED IN THE Y REGISTER   *
                        01460 *                                              *
                        01470 ************************************************
7098 8B    30           01480 COUT    ADDA    #$30    *CONVERT TO ASCII
709A AD    9F A002      01490         JSR     [COUT1] *OUTPUT ASCII CHARACTER
709E 39                 01500         RTS             *AND RETURN
           0000         01510         END

00000 TOTAL ERRORS
```

```
'*************************************************
'*                                               *
'*                PROGRAM MAIN                   *
'*           Version 1.0, Sep 85                 *
'*                                               *
'*  THIS PROGRAM IS THE MAIN HUB OF THE PCM      *
'*  MONITOR SYSTEM.                              *
'*************************************************

10 REM PROGRAM MAIN
20 CLS
30 REM PRINT MAIN MENU
40 PRINT@0, STRING$(32,"*");
50 PRINT"*                                      *";
60 PRINT"*            MAIN MENU                 *";
70 PRINT"*                                      *";
80 PRINT STRING$ (32,"*")
90 PRINT"  F)RAME SPECIFICATION"
100 PRINT"  P)ARAMETER SPECIFICATIONS"
110 PRINT"  D)ISPLAY FORMATS"
120 PRINT"  X)ECUTE TO BEGIN PCM MONITOR"
130 PRINT"  S)ETUP NEW SYSTEM"
140 PRINT"  Q)UIT"
150 PRINT
160 REM GET INPUT
170 PRINT"   WHICH ONE? -->";:INPUT A$
180 IF A$="P" THEN RUN "PSETUP"
190 IF A$="D" THEN RUN "DSETUP"
200 IF A$="F" THEN RUN "FSETUP"
210 IF A$="S" THEN 340
220 IF A$="X" THEN RUN "EXECUTE"
230 IF A$="Q" THEN CLS:PRINT"BYE!":END
240 GOTO40
250 REM ************************
260 REM *                      *
270 REM *    SETUP ROUTINE     *
280 REM *                      *
290 REM * SETS RECORD NUMBER   *
300 REM * STORED TO 0 IN BOTH  *
310 REM * PARAM & DISPLAY.DAT  *
297 REM *                      *
320 REM ************************
330 REM
340 CLS:PRINT STRING$(32,"*");
350 PRINT"*                                      *";
360 PRINT"*       SETUP NEW SYSTEM               *";
370 PRINT"*                                      *";
380 PRINT STRING$(32,"*");
390 PRINT
400 PRINT"warning warning  warning warning";
410 PRINT
420 PRINT"THIS PROGRAM WILL DELETE YOUR"
```

```
430 PRINT"DISPLAY FILES.  BE SURE YOU'VE"
440 PRINT"SAVED THEM TO ANOTHER FILENAME."
450 PRINT
460 PRINT"DO YOU WISH TO CONTINUE OR ABORT";
470 INPUT"PRESS C OR A ->";A$
480 IF A$<>"C" THEN RUN
490 OPEN "D", #1, "PARAM.DAT", 60
500 WRITE #1,"0"
510 PUT #1,201
520 CLOSE #1
530 OPEN "D", #1, "DISPLAY.DAT", 250
540 WRITE #1,"0"
550 PUT #1,101
560 CLOSE #1
570 RUN
```

```
10 REM PROGRAM FRAME SETUP
20 CLEAR 2000:REM MAKE STRING SPACE
30 GOTO 1960:REM START OF PROGRAM
40 REM  ************************
50 REM  * FRAME SET UP PROGRAM *
55 REM  * Version 1.0, Sep 85  *
60 REM  *                      *
70 REM  *  THIS PROGRAM ALLOWS *
80 REM  *  THE USER TO INPUT   *
90 REM  *  FRAME SPECIFICATIONS*
100 REM  * INTO THE PCM MONITOR*
110 REM  ************************
120 '
130 '
140 '
150 REM ************************
160 REM *                      *
170 REM * VARIABLE DEFINTIONS  *
180 REM *                      *
260 REM *                      *
270 REM ************************
280 '
290 '
300 ' NAME$=NAME OF DATA FILE TO ACCESS
310 ' REC = RECORD NUMBER IN NAME$ TO ACCESS
320 ' TY$=TYPE OF PCM: NRZ-L OR BI-PHASE-L
330 ' TY=VARIBLE TO SHOW WHAT PCM TYPE IS, 1=NRZ, 2=BI-PHASE
340 ' RA$=PCM RATE: AUTO OR VALUE
350 ' RA=VARIABLE TO SHOW BIT RATE, AUTO OR NUMBER
360 ' O$=PCM ORDER: MSB FIRST OR LSB FIRST
370 ' O=VARIABLE TO SHOW WHAT PCM ORDER IS, 1=MSB, 2=LSB
    FIRST
380 ' PO$=PCM POLARITY: NORMAL OR INVERSE
390 ' PO=VARIABLE TO SHOW WHAT POLARITY PCM IS. 1=NORMAL,
    2=INVERSE
400 ' SW$=# SYNC WORDS
410 ' SW$(1-3)=STRING FOR SYNC WORD 1-3
420 ' I, J, K = COUNTER VARIABLES
430 ' X=GENERAL PURPOSE RESULT VARIABLE
440 ' TIME=CURSOR FLASH RATE
450 ' SP(I)=SCREEN POSITION FOR DISPLAY LINE I
460 ' VP(I)=SCREEN POSITION FOR NON-BINARY DATA IN LINE I
470 ' SAME=USED IN GET ROUTINE TO TELL IF KEY WAS PRESSED
480 ' PSN=POSITION POINTER USED IN LINE INPUT ROUTINE
490 ' CH=CHARACTER READ FROM SCREEN IN GET ROUTINE
500 ' KY=VALUE OF KEY USED IN GET ROUTINE
510 ' KY$=STRING OF KEY INPUT
520 ' SIZE=SIZE OF DATA RECORD. PARAM.DAT=60,
    DISPLAY.DAT=250
530 ' W$()=#WORDS=FRAMELENGTH
540 ' B$=STRING CONTANING BITS PER WORD
550 ' B=VALUE OF B$
```

```
560 ' N=NUMBER OF CHARACTERS USED IN INPUT LINE ROUTINE
570 ' TP=TYPE POINTER, POINTS TO DISPLAY TYPE
580 ' PAD=CHARACTER SPACES TO MAKE SW$()=16
590 ' LN=POINTER FOR LINE NUMBER
600 '
610 '
620 '
630 REM ***************************
640 REM *                         *
650 REM *      INITIALIZATION     *
660 REM *        SUBROUTINE       *
670 REM *                         *
680 REM ***************************
690 REM
700 POKE 65495,0:REM SET FAST SPEED
710 TIME=50:REM CURSOR TIME DELAY
720 '
730 REM SETUP SCREEN POSITIONS FOR THE INPUT ROUTINES
740 SP(1)=142:SP(2)=173:SP(3)=206:SP(4)=237:SP(5)=270
745 SP(6)=302:SP(7)=336:SP(8)=366:SP(9)=398:SP(10)=430
750 '
760 REM SETUP DEFAULTS
770 TY=1:RA=1:O=1:PO=1:RA$="auto":SW$="3":SW$(1)="
                  "
775 SW$(2)="                    ":SW$(3)="
                ":B$="10":W$="100"
780 TY$(0)="AUTO NRZ BI-PHASE"
790 TY$(1)="auto NRZ BI-PHASE"
800 TY$(2)="AUTO nrz BI-PHASE"
810 TY$(3)="AUTO NRZ bi-phase"
820 O$(0)="LSB MSB"
830 O$(1)="lsb MSB"
840 O$(2)="LSB msb"
850 PO$(0)="NORMAL INVERSE"
860 PO$(1)="normal INVERSE"
870 PO$(2)="NORMAL inverse"
880 RETURN
890 '
900 '
910 REM HEADER OUT SUBROUTINE
920 CLS: PRINT STRING$(32,"*");
930 PRINT "*";
940 FOR I=1 TO (30-LEN(A$))/2:PRINT" ";:NEXT I
950 PRINT A$;
960 IF POS(0)=31 THEN 980
970 PRINT " ";:GOTO960
980 PRINT "*";
990 PRINT STRING$(32,"*");
1000 RETURN
1010 '
1020 '
1030 REM  ************************
```

```
1040 REM  *                          *
1050 REM  *      LINE INPUT          *
1055 REM  *       SUBROUTINE         *
1060 REM: *                          *
1070 REM: ************************
1080 '
1090 SAME=0:PSN=1:PRINT@SP(LN),"["+A$;:PRINT@SP(LN)+N+1,"]";
1100 REM MAKE FLASHING CURSOR
1110 IF PSN=N+1 THEN 1160:REM END OF LINE
1120 X=SP(LN)+PSN+1024:REM DIRECT READ OF SCREEN
1130 CH=PEEK(X)
1140 POKE X,109:FOR I=1 TO 2*TIME:NEXT I:POKE X,CH:FOR I=1
     TO TIME:NEXT I
1150 REM 109 = "-"
1160 A$=INKEY$: IF A$="" THEN 1110
1170 KY=ASC(A$)
1180 IF KY=12 OR KY=13 THEN 1270:REM CLEAR OR ENTER
1190 IF KY=8 AND PSN =1 THEN 1160:REM LEFT ARROW
1200 IF KY=8 THEN PSN=PSN-1:GOTO1160:REM LEFT ARROW
1210 IF KY=10 OR KY=94 THEN 1270:REM DOWN OR UP
1220 IF PSN <1 OR PSN >N THEN 1160 ELSE PRINT@SP(LN)+PSN,
     A$;
1230 IF KY=9 THEN 1240 ELSE SAME=1:REM NEW DATA ENTERED
1240 PSN=PSN+1
1250 GOTO1160
1260 REM COLLECT CHARACTERS INTO A$
1270 A$=""
1280 FOR PSN=1 TO N:X=PEEK(SP(LN)+PSN+1024):IF X>95 THEN
     X=X-64
1290 A$=A$+CHR$(X):NEXT PSN
1300 PRINT@SP(LN)," ";:PRINT@SP(LN)+N+1," ";:REM REMOVE
     BRACKETS
1310 RETURN
1320 REM
1330 REM
1340 REM ************************
1350 REM *                      *
1360 REM *  DISK INPUT ROUTINE  *
1370 REM *                      *
1380 REM *  RECORD # IN PN      *
1390 REM *  RECORD NAME IN NAME$*
1400 REM *  RETURNS DATA IN A$  *
1410 REM *                      *
1420 REM ************************
1430 IF REC=0 THEN RETURN
1440 OPEN"D", #1, NAME$,SIZE
1450 GET#1, REC
1460 INPUT #1,A$
1470 CLOSE #1
1480 RETURN
1490 REM ************************
1500 REM *                      *
```

```
1510 REM *  DISK OUTPUT ROUTINE *
1520 REM *                      *
1530 REM *  RECORD # IN PN      *
1540 REM *  RECORD NAME IN NAME$*
1550 REM *  DATA IN A$          *
1560 REM *                      *
1570 REM ************************
1580 IF REC=0 THEN RETURN
1590 OPEN "D", #1, NAME$, SIZE
1600 WRITE #1,A$
1610 PUT #1,REC
1620 CLOSE #1
1630 RETURN
1640 REM
1650 REM ************************
1660 REM *                      *
1670 REM * CONVERT A$ TO SPEC   *
1675 REM *     SUBROUTINE       *
1680 REM *                      *
1690 REM ************************
1700 REM
1710 B$=LEFT$(A$,2):W$=MID$(A$,3,3):TY$=MID$(A$,6,8)
1715 RA$=MID$(A$,14,4):O$=MID$(A$,18,3):PO$=MID$(A$,21,7)
1720 SW$=MID$(A$,28,1): SW$(1)=MID$(A$,29,16):
     SW$(2)=MID$(A$,45,16)
1725 SW$(3)=MID$(A$,61,16)
1730 TY=1:IF TY$="NRZ      " THEN TY=2
1740 IF TY$="BI-PHASE" THEN TY=3
1750 IF RA$<>"auto" THEN RA=2 ELSE RA=1
1760 IF O$="MSB" THEN O=2 ELSE O=1
1770 IF PO$="INVERSE" THEN PO=2 ELSE PO=1
1780 RETURN
1781 '
1782 '
1783 REM ************************
1784 REM *                      *
1785 REM *    TOGGLE INPUT      *
1786 REM *     SUBROUTINE       *
1787 REM *                      *
1788 REM ************************
1790 PRINT@SP(LN),A$(0):FOR I=1 TO TIME:NEXT I
1800 PRINT@SP(LN),A$(J):FOR I=1 TO TIME:NEXT I:A$=INKEY$:IF
     A$="" THEN 1790
1810 KY=ASC(A$)
1815 REM CHECK FOR UP, DOWN, CLEAR, OR ENTER KEYS
1820 IF KY=94 OR KY=10 OR KY=12 OR KY=13 THEN RETURN
1830 IF KY=8 THEN J=J-1 ELSE J=J+1
1840 IF J>N THEN J=1
1850 IF J<1 THEN J=N
1860 GOTO 1790
1870 REM
1880 REM
```

```
1890 REM ***********************
1900 REM *                     *
1910 REM *  START OF PROGRAM   *
1920 REM *                     *
1930 REM ***********************
1940 '
1950 '
1960 GOSUB 700:REM INIT
1970 A$="FRAME SPECIFICATION PAGE":GOSUB 920: REM OUTPUT
     HEADER
1980 PRINT
1990 PRINT"C)REATE NEW FRAME SPECIFICATION";
2000 '
2010 PRINT:PRINT"L)OAD FRAME SPECIFICATION"
2020 PRINT "E)DIT SPECIFICATION IN MEMORY"
2030 PRINT"Q)UIT"
2040 PRINT
2050 INPUT "WHICH ONE ->";A$
2060 IF A$="L" THEN 2180
2070 IF A$="E" THEN 2410
2080 IF A$="C" THEN 2310
2090 IF A$="Q" THEN RUN"MAIN"
2100 GOTO 1970:REM NOT ANY THEN TRY AGAIN
2110 '
2120 '
2130 REM ************************
2140 REM *                      *
2150 REM *        LOAD          *
2160 REM *                      *
2170 REM ************************
2180 A$="LOADING FRAME SPECIFICATION":GOSUB 920:REM OUTPUT
     HEADER
2190 NAME$="DISPLAY.DAT":SIZE=250:REC=102:REM 102 HAS FRAME
     SPEC
2200 GOSUB 1430:REM READ SPEC IN A$
2210 GOSUB 1710:REM CONVERT A$
2220 A$="CURRENT FRAME SPECIFICATION":REM SET CREATE HEADER
2230 GOTO 2420:REM NOW EDIT IT
2240 '
2250 '
2260 REM ************************
2270 REM *                      *
2280 REM *       CREATE         *
2290 REM *                      *
2300 REM ************************
2310 GOSUB 700:REM RE-INIT AND SET DEFAULTS
2320 A$="CREATE FRAME SPECIFICATION":REM OUTPUT HEADER
2330 GOTO 2420: NOW FILL IN VARIABLES AND EDIT
2340 '
2350 '
2360 REM ************************
2370 REM *                      *
```

```
2380 REM *         EDIT          *
2390 REM *                       *
2400 REM *************************
2410 A$="EDIT FRAME SPECIFICATION"
2420 GOSUB 920:REM OUTPUT HEADER
2430 PRINT@448,STRING$(32,"*");
2440 PRINT@480," USE NUMBERS, ARROWS, OR CLEAR";
2450 PRINT@128,"BITS/WORD       ";B$;"  MAX IS 16"
2460 PRINT "WORDS/FRAME    ";W$;"  MAX IS 200"
2470 PRINT "PCM TYPE       ";TY$(TY)
2480 PRINT "PCM BIT RATE   ";RA$;" KHZ A)UTO"
2490 PRINT "PCM ORDER      ";O$(O)
2500 PRINT "PCM POLARITY   ";PO$(PO)
2510 PRINT"# OF SYNC WORDS  ";SW$;"  MAX # IS 3"
2520 PRINT"SYNC DEF->";
2530 PRINT@367,SW$(1)
2540 PRINT@399,SW$(2)
2550 PRINT@431,SW$(3);
2560 LN=1
2570 GOSUB 2700: REM EDIT SUBROUTINE
2580 CLS:PRINT"SAVING NEW FRAME SPECIFICATIONS"
2590 NAME$="DISPLAY.DAT":SIZE=250:REC=102
2595 REM NOW BUILD STRING AND SAVE IT
2600 A$=B$+W$+TY$+RA$+O$+PO$+SW$+SW$(1)+SW$(2)+SW$(3)
2610 GOSUB 1580: REM SAVE TO DISK
2620 GOTO 1970:REM START OVER AGAIN
2630 '
2640 '
2650 REM *************************
2660 REM *                       *
2670 REM *         EDIT          *
2675 REM *     SUBROUTINE        *
2680 REM *                       *
2690 REM *************************
2700 IF LN<>1 THEN 2760
2705 REM NOW GET LINE FROM LINE INPUT ROUTINE
2710 N=2:A$=B$:GOSUB 1090:B$=A$:IF VAL(B$)>16 OR VAL(B$)<1
     THEN 2710
2720 IF VAL(B$)=16 THEN 2740:REM DON'T PAD
2725 REM CREATE PAD TO MAKE LENGTH OF SW$()=16
2730 PAD$="":FOR I=1 TO 16-VAL(B$):PAD$=PAD$+" ":NEXT I
2735 REM CLEAR SYNC DATA IF SIZE CHANGES
2740 IF SAME=1 THEN PRINT@SP(8),"":PRINT@SP(9),"":
     PRINT@SP(10), "
                   ";:SW$(1)="":SW$(2)="":SW$(3)=""
2750 GOTO 2930
2760 IF LN=2 THEN N=3:A$=W$:GOSUB 1090:W$=A$:IF VAL(W$)<1
     THEN 2760 ELSE 2930
2770 IF LN=3 THEN  N=3: J=TY: A$(0)=TY$(0): A$(1)=TY$(1):
     A$(2)=TY$(2): A$(3)=TY$(3): GOSUB 1790: TY=J:GOTO 2930:
     REM TOGGLE INPUT
2780 IF LN<>4 THEN 2820
```

```
2790 N=4:A$=RA$:GOSUB 1090:RA$=A$
2800 IF VAL(A$)=0 THEN RA$="auto":PRINT@SP(4)+1,RA$;:
     GOTO 2930
2810 IF VAL(RA$)>100 THEN 2780
2820 IF LN=5 THEN N=2: J=O: A$(0)=O$(0): A$(1)=O$(1):
     A$(2)=O$(2): GOSUB 1790:O=J:GOTO 2930:REM TOGGLE INPUT
2830 IF LN=6 THEN N=2: J=PO: A$(0)=PO$(0): A$(1)=PO$(1):
     A$(2)=PO$(2): GOSUB 1790:PO=J:GOTO 2930:REM TOGGLE
     INPUT
2840 IF LN<>7 THEN 2900
2850 N=1:A$=SW$:GOSUB1090:SW$=A$:REM DISK INPUT
2860 IF VAL(SW$)>3 THEN 2850
2870 IF VAL(SW$)=1 THEN SW$(2)="
                ":PRINT@SP(9),SW$(2)
2880 IF VAL(SW$)=1 OR VAL(SW$)=2 THEN SW$(3)="
      ":PRINT@SP(10),SW$(3)
2890 GOTO 2930
2900 IF LN=8 THEN N=VAL(B$):A$=SW$(1):GOSUB 1090:
     SW$(1)=A$+PAD$:GOTO 2930:REM MUST PAD SW$(N) TO = 16
     CHAR FOR SAVING
2910 IF LN=9 AND VAL(SW$)>1 THEN N=VAL(B$):A$=SW$(2):GOSUB
     1090: SW$(2)=A$+PAD$:GOTO2930
2920 IF LN=10 AND VAL(SW$)>2 THEN N=VAL(B$): A$=SW$(3):
     GOSUB 1090: SW$(3)=A$+PAD$: GOTO 2930
2930 IF KY=12 THEN 2990:REM CLEAR SETS VARIABLES AND SAVES
     THEM
2940 LN=LN+1:REM ADVANCE LINE
2950 IF KY=94 THEN LN=LN-2:REM MOVE BACK ONE LINE
2960 IF LN<1 THEN LN=10:REM WRAP AROUND
2970 IF LN>10 THEN LN=1
2980 GOTO 2700
2990 IF TY=1 THEN TY$="AUTO     "
3000 IF TY=2 THEN TY$="NRZ      "
3010 IF TY=3 THEN TY$="BI-PHASE"
3020 IF O=2 THEN O$="MSB" ELSE O$="LSB"
3030 IF PO=2 THEN PO$="INVERSE" ELSE PO$="NORMAL "
3040 RETURN
```

```
10 REM ***********************************
20 REM *                                 *
30 REM *         PROGRAM PSETUP          *
40 REM *                                 *
50 REM * THIS PROGRAM ALLOWS THE USER *
60 REM * TO INPUT DATA WORD SPECIFI-  *
70 REM * CATIONS INTO THE PCM MONITOR *
80 REM * WORDS IN THE FILE "PARAM.DAT *
90 REM *                                 *
100 REM ***********************************
110 REM
120 REM
130 REM ***************************
140 REM *                         *
150 REM *   VARIABLES USED IN     *
160 REM *         PSETUP          *
170 REM *                         *
180 REM ***************************
190 '
200 ' MR$=MAX NUMBER OF RECORDS STORED IN DISPLAY.DAT
210 ' NAME$=NAME OF DATA FILE TO ACCESS
220 ' REC = RECORD NUMBER IN NAME$ TO ACCESS
230 ' A$=GENERAL PURPOSE STRING VARIABLE
240 ' R$=ALTERNATE FOR MAX RECORDS STORAGE
250 ' SV$=GENERAL PURPOSE SAVE STRING VARIABLE
260 ' SV=GENERAL PURPOSE SAVE VALUE VARIABLE
270 ' I, J = COUNTER VARIABLES
280 ' X=GENERAL PURPOSE RESULT VARIABLE
290 ' TIME=CURSOR FLASH RATE
300 ' DLAY=DELAY FOR WAIT LOOPS
310 ' PSN=POSITION POINTER USED IN LINE INPUT ROUTINE
320 ' CH=CHARACTER READ FROM SCREEN IN GET ROUTINE
330 ' KY=VALUE OF KEY USED IN GET ROUTINE
340 ' PN$=DATA WORD NUMBER
350 ' N$=PARAMETER NAME
360 ' FP$=PARAMETER NUMBER
370 ' U$=PARAMETER UNITS
380 ' MN$=MINIMUM VALUE FOR PARAMETER
390 ' MX$=MAXIMUM VALUE OF PARAMETER
400 ' AL$=ALARM LOW VALUE
410 ' AH$=ALARM HIGH VALUE
420 ' D$()=DISPLAY TYPE, DECIMAL, HEX ETC
430 ' W$()=#WORDS=FRAMELENGTH
440 ' LN=SCREEN LINE NUMBER USED IN LINE INPUT ROUTINE
450 ' N=NUMBER OF CHARACTERS USED IN LINE INPUT ROUTINE
460 '
470 REM PROGRAM PSETUP
480 GOTO1890:REM START OF PROGRAM
490 REM ***************************
500 REM *                         *
510 REM *     INITIALIZATION      *
520 REM *     SUBROUTINE          *
```

```
530 REM *                          *
540 REM **************************
550 REM
560 POKE 65495,0:REM SET FAST SPEED
570 DIM SP(16)
580 DLAY=500:REM DELAY FOR WAIT LOOPS
590 FOR I=1TO16:READ SP(I):NEXTI
600 DATA 0,0,0,0,147,179,211,241,273,305,337,369,0,0,0,0
610 N$="        ":REM SET DEFAULTS
620 FP$="    ":U$="      ":MN$="       ":MX$="       ": AL$="
        ":AH$="       "
630 RETURN
640 REM
650 REM  ************************
660 REM  *                        *
670 REM  * LINE INPUT ROUTINE     *
680 REM: * N = # OF CHARACTERS    *
690 REM: * SP= SCREEN POSITION    *
700 REM: * RETURNS STRING IN A$   *
710 REM: * N = MAX CHARACTERS     *
720 REM: * I = TEMP COUNTER       *
730 REM: * PSN = LOCAL POSITION   *
740 REM  * LN= LINE ON SCREEN     *
750 REM: *                        *
760 REM: ************************
770 REM
780 PSN=1:PRINT@SP(LN),"["+A$;:PRINT@SP(LN)+N+1,"]";
790 REM MAKE FLASHING CURSOR
800 IF PSN=N+1 THEN 850
810 X=SP(LN)+PSN+1024
820 CH=PEEK(X)
830 POKE X,109:FOR I=1TO40:NEXT I:POKE X,CH:FOR I=1TO20:NEXT
I
840 REM 109 = "-"
850 A$=INKEY$: IF A$="" THEN 800
855 KY=ASC(A$):REM GET RETURN KEY
860 IF KY=12 THEN QUIT=1:GOTO 950:REM CLEAR KEY
870 IF KY=13 THEN 950
880 IF KY = 8 AND PSN =1 THEN 850:REM LEFT ARROW BUT AT LEFT
    SIDE ALREADY
890 IF KY = 8 THEN PSN=PSN-1:GOTO850:REM LEFT ARROW TO  BACK
UP
900 IF KY=10 OR KY=94 THEN 950:REM USE UP AND DOWN ARROWS
910 IF PSN <1 OR PSN >N THEN 850 ELSE PRINT@SP(LN)+PSN, A$;
920 PSN=PSN+1
930 GOTO850
940 REM COLLECT CHARACTERS INTO A$
950  A$="":FOR  PSN=1  TO  N:X=PEEK(SP(LN)+PSN+1024):IF X>95
THEN
    X=X-64
960 A$=A$+CHR$(X):NEXT PSN
970  PRINT@SP(LN),"  ";:PRINT@SP(LN)+N+1,"   ";:REM   REMOVE
```

```
BRACKETS
980 RETURN
990 REM
1000 REM
1010 REM ************************
1020 REM *                      *
1030 REM *  DISK INPUT ROUTINE  *
1040 REM *                      *
1050 REM *  RECORD # IN PN      *
1060 REM *  RECORD NAME IN NAME$*
1070 REM *  RETURNS DATA IN A$  *
1080 REM *                      *
1090 REM ************************
1100 OPEN "D", #1, NAME$,60
1110 GET#1, REC:REM 201 FOR #RECORDS
1120 INPUT #1,A$
1130 CLOSE #1
1140 RETURN
1150 REM ************************
1160 REM *                      *
1170 REM *  DISK OUTPUT ROUTINE *
1180 REM *                      *
1190 REM *  RECORD # IN PN      *
1200 REM *  RECORD NAME IN NAME$*
1210 REM *  DATA IN A$          *
1220 REM *                      *
1230 REM ************************
1240 OPEN "D", #1, NAME$, 60
1250 WRITE #1,A$
1260 PUT #1,REC
1270 CLOSE #1
1280 RETURN
1290 REM
1300 REM
1310 REM
1320 REM ************************
1330 REM *                      *
1340 REM *     SUBROUTINE       *
1350 REM *   LIST DATA WORDS    *
1360 REM *                      *
1370 REM ************************
1380 CLS:PRINT"DATA WORD LISTING OF    RECORDS";
1390 PRINT@32,"**********************************";
1400 PRINT"      WORD #     NAME  PARM"
1410 PRINT
1420 REC=201:GOSUB 1100
1430 R$=A$
1440 PRINT@20,R$;
1450 PRINT@128,"";
1460 FOR I=1 TO VAL(R$)
1470 REC = I
1480 GOSUB 1100:REM READ IN RECORD
```

```
1490 PRINT"       ";I,LEFT$(A$,7);" ";MID$(A$,8,3)
1500 IF INT(I/10)=I/10 THEN 1520
1510 NEXT I
1520 PRINT:INPUT"NUMBER=SEL, CR=CONT, E=EXIT ";A$
1530 IF A$="E" THEN 1600
1540 IF VAL(A$)<>0 THEN 1600
1550 IF I=>VAL(R$) THEN 1600
1560 CLS:PRINT"DATA WORD LISTING OF ";R$;" RECORDS";
1570 PRINT"**********************************";
1580 PRINT"      WORD #        NAME":PRINT
1590 GOTO 1510
1600 RETURN
1610 '
1620 REM *********************************
1630 REM *                               *
1640 REM *          SUBROUTINE           *
1650 REM *   DISPLAY DEFINITION PAGE     *
1660 REM *                               *
1670 REM *********************************
1680 CLS:PRINT"   DATA WORD DEFINITION PAGE   "
1690 PRINT"**********************************";
1700 PRINT A$;PN
1710 PRINT
1720 PRINT"PARAMETER NAME -->  ";N$
1730 PRINT"PARAMETER NUMBER->  ";FP$
1740 PRINT"PARAMETER UNITS ->  ";U$
1750 PRINT
1760 PRINT"RANGE MINIMUM ->  ";MN$
1770 PRINT"RANGE MAXIMUM ->  ";MX$
1780 PRINT"ALARM MINIMUM ->  ";AL$
1790 PRINT"ALARM MAXIMUM ->  ";AH$
1800 PRINT
1810 PRINT"**********************************";
1820 PRINT"ENTER FOR NEXT, CLEAR TO ACCEPT";
1830 RETURN
1840 REM *************************
1850 REM *                       *
1860 REM *    START OF PROGRAM   *
1870 REM *                       *
1880 REM *************************
1890 GOSUB 560:REM INIT
1900 REM PN=DATA WORD NUMBER (3 SPACES)
1910 REM N$=PARAMETER NAME (7 SPACES)
1920 REM FP$=PARAMATER NUMBER (3 SPACES)
1930 REM U$=PARAMETER UNITS (5 SPACES)
1940 REM MN$=MIN VALUE (6 SPACES)
1950 REM MX$=MAX VALUE (6 SPACES)
1960 REM AL$=ALARM LOW VALUE (6 SPACES)
1970 REM AH$=ALARM HIGH VALUE (6 SPACES)
1980 CLS
1990 NAME$="PARAM.DAT"
2000 CLS
```

```
2010 PRINT STRING$(32,"*");
2020 PRINT"*";STRING$(30," ");"*";
2030 PRINT"*  DATA WORD DEFINITION PAGE    *";
2040 PRINT"*";STRING$(30," ");"*";
2050 PRINT STRING$ (32,"*");
2060 PRINT @192,"A)DD NEW DATA WORD"
2070 PRINT"E)DIT DATA WORD"
2080 PRINT"I)NSERT DATA WORD"
2090 PRINT"D)ELETE DATA WORD"
2100 PRINT"L)IST PARAMETER TABLE"
2110 PRINT"Q)UIT"
2120 PRINT:INPUT"WHICH ONE";KY$
2130 IF KY$="A" THEN 2250
2140 IF KY$="Q" THEN RUN"MAIN"
2150 IF KY$="E" THEN 3150
2160 IF KY$="D" THEN 3340
2170 IF KY$="I" THEN 3580
2180 IF KY$="L" THEN GOSUB 1380
2190 GOTO 2000
2200 REM ************************
2210 REM *                      *
2220 REM *      ADD DATA WORD   *
2230 REM *                      *
2240 REM ************************
2250 REC=201:REM DATA WORD COUNT IN 201
2260 GOSUB 1100:REM READ # RECORDS IN FILE
2270 PN=VAL(A$)+1
2280 A$="ADDING DATA WORD #"
2290 GOSUB 2430:REM GET NEW DATA
2300 IF A$="T" THEN 2280:REM TRY AGAIN
2310 PRINT:PRINT"UPDATING # DATA WORDS RECORD"
2320 REC=201:A$=STR$(PN)
2330 GOSUB 1240
2340  RUN:  REM  START AGAIN WITH MAIN MENU.   CLEARS STRING
SPACE
2350 REM
2360 REM
2370 REM ************************
2380 REM *                      *
2390 REM *    RETRIEVE DATA     *
2400 REM *     SUBROUTINE       *
2410 REM *                      *
2420 REM ************************
2430 TEMP$=A$:REM SAVE TYPE OF PAGE WE'RE IN
2440 GOSUB 1680:REM OUTPUT DISPLAY PAGE
2450 QUIT=0:DS=1:REM SET DEFAULT DISPLAY TYPE = INT
2460 LN=5:N=7:A$=N$:GOSUB780:REM GET NAME
2470 N$=A$
2480 IF QUIT=1 THEN 2860:REM QUIT THIS ENTRY
2490 IF KY=94 THEN 2780:REM UP
2500 LN=6:N=3:A$=FP$:GOSUB780:REM GET PARAMETER NUMBER
2510 FP$=A$
```

```
2520 IF QUIT=1 THEN 2860
2530 IF KY=94 THEN 2460
2540 LN=7:N=5:A$=U$:GOSUB 780:REM GET UNITS
2550 U$=A$
2560 IF QUIT=1 THEN 2860
2570 IF KY=94 THEN 2500
2580 '
2590 ' GET MINIMUM OF RANGE
2600 LN=9:N=6:A$=MN$:GOSUB 780
2610 MN$=A$
2620 IF QUIT=1 THEN 2860
2630 IF KY=94 THEN 2540
2640 '
2650 ' GET MAXIMUM OF RANGE
2660 LN=10:N=6:A$=MX$:GOSUB 780
2670 MX$=A$
2680 IF QUIT=1 THEN 2860
2690 IF KY=94 THEN 2600
2700 '
2710 ' GET ALARM LOW
2720 LN=11:N=6:A$=AL$:GOSUB 780
2730 AL$=A$
2740 IF QUIT=1 THEN 2860
2750 IF KY=94 THEN 2660
2760 '
2770 ' GET ALARM HIGH
2780 LN=12:N=6:A$=AH$:GOSUB 780
2790 AH$=A$
2800 IF QUIT=1 THEN 2860
2810 IF KY=94 THEN 2720
2820 GOTO2460
2830 '
2840 '
2850 REM CHECK ERRORS AND PRINT SAVE SCREEN
2860 CLS:PRINT:PRINT
2870 QUIT=0:REM RESET QUIT BECAUSE WE AREN'T QUITTING
2880 A$=TEMP$:REM RETURN TYPE OF PAGE IF NEEDED FOR ERROR
2890  IF   VAL(AH$)>VAL(MX$)   THEN   PRINT"ALARM   OUT   OF
LIMITS":FOR
     I=1TO DLAY:NEXT I:GOSUB 1680:GOTO2780
2900   IF   VAL(AL$)<VAL(MN$)   THEN   PRINT"ALARM   OUT   OF
LIMITS":FOR I=1TO DLAY:NEXT I:GOSUB 1680:GOTO 2720
2910 PRINT"S)AVE DATA WORD #";PN
2920 PRINT"T)RY AGAIN?"
2930 PRINT"R)ETURN TO MENU?"
2940 PRINT:INPUT"WHICH ONE>";A$
2950 IF A$="S" THEN 3030:REM SAVE RECORD
2960 IF A$="T" THEN 2990:REM RETURN WITHOUT SAVING
2970 IF A$="R" THEN RUN
2980 GOTO 2860
2990 RETURN:REM A$ HAS RETURN CONDITIONS
3000 '
```

```
3010 '
3020 REM BUILD SAVING STRING AND DO IT
3030 A$=N$+FP$+U$+MN$+MX$+AL$+AH$
3040 REC=PN
3050 GOSUB 1240:REM SAVE DATA WORD
3060 PRINT:PRINT"DATA WORD #";PN;N$;" SAVED"
3070 FOR I=1TO4*DLAY:NEXT I
3080 A$="S":REM SHOW CALLING PROGRAM THAT REC WAS SAVED
3090 RETURN
3100 REM ************************
3110 REM *                      *
3120 REM *    EDIT DATA WORD    *
3130 REM *                      *
3140 REM ************************
3150   CLS:PRINT@484,"PRESS   L   TO   LIST,   E   TO   EXIT
":PRINT@35,"ENTER
     DATA WORD TO EDIT ";:INPUT A$
3160 IF A$="L" THEN GOSUB 1380
3170 IF A$="E" THEN 3280:REM DON'T EDIT
3180 PN=VAL(A$): IF PN=0 THEN 3150:REM MUST BE NUMBER
3190 REC=201:GOSUB 1100
3200  IF  PN>VAL(A$)  THEN PRINT"YOU ONLY HAVE";A$;" RECORDS
SAVED":FOR I=1TO2000:NEXT I:GOTO3150
3210  IF  PN>200   THEN   PRINT"MAX   NUMBER   IS   200":FOR
I=1TO2000:NEXT
     I:GOTO3150
3220 REC=PN
3230 GOSUB 1100:REM GET OLD DATA
3240 N$=LEFT$(A$,7): FP$=MID$(A$,8,3): U$=MID$(A$,11,5):
     MN$=MID$(A$,16,6):                    MX$=MID$(A$,22,6):
AL$=MID$(A$,28,6):
     AH$=MID$(A$,34,6)
3250 A$="EDITING DATA WORD # "
3260 GOSUB 2430
3270 IF A$="T" THEN 3250
3280 RUN: REM USE RUN TO CLEAR STRING SPACE
3290 REM ************************
3300 REM *                      *
3310 REM *  DELETE DATA WORD    *
3320 REM *                      *
3330 REM ************************
3340   CLS:PRINT@484,"PRESS   L   TO   LIST,   E   TO   EXIT
":PRINT@32,"ENTER DATA WORD TO DELETE ";:INPUT A$
3350 IF A$="L" THEN GOSUB 1380
3360 IF A$="E" THEN 3570
3370 PN=VAL(A$): IF PN=0 THEN 3340:REM MUST BE A NUMBER
3380 REC=201:GOSUB 1100
3390 IF PN>VAL(A$) THEN PRINT"YOU ONLY HAVE";A$;" RECORDS
     SAVED": FOR I=1 TO 2000: NEXT I: GOTO 3340
3400  IF  PN>200  THEN  PRINT"MAX  NUMBER IS 200":FOR I=1 TO
2000:NEXT I:GOTO 3340
3410 REC=PN
```

```
3420 SV$=A$:REM SAVE # RECORDS IN FILE
3430 GOSUB 1100:REM GET OLD DATA
3440 N$=LEFT$(A$,7): FP$=MID$(A$,8,3): U$=MID$(A$,11,5):
     MN$=MID$(A$,16,6):                    MX$=MID$(A$,22,6):
AL$=MID$(A$,28,6):
     AH$=MID$(A$,34,6)
3450 A$="DELETE DATA WORD # "
3460 GOSUB 1680
3470 PRINT@448,"DELETE THIS DATA WORD? Y/N  ";:INPUT A$
3480 IF A$<>"Y" THEN RUN
3490 CLS:PRINT"     DELETING DATA WORD";PN
3500 FOR I=PN+1 TO VAL(SV$)
3510 REC=I:GOSUB 1100:REM READ OLD RECORD
3520 REC=I-1:GOSUB 1240:REM PUT IN NEXT LOWER POSITION
3530 NEXT I
3540 REC=201:A$=STR$(VAL(SV$)-1):REM UPDATE # RECORDS STORED
3550 GOSUB 1240
3560 RUN
3570 RUN
3580 REM
3590 REM ************************
3600 REM *                      *
3610 REM *   INSERT DATA WORD   *
3620 REM *                      *
3630 REM ************************
3640 REM
3650   CLS:PRINT@484,"PRESS   L   TO   LIST,   E   TO   EXIT
":PRINT@32,"ENTER
     WORD# TO BE INSERTED";:INPUT A$
3660 IF A$="L" THEN GOSUB 1380
3670 IF A$="E" THEN RUN:REM EXIT
3680 PN=VAL(A$): IF PN=0 THEN  3650
3690 REC=201:GOSUB1100
3700 SV=VAL(A$):REM SAVE TOTAL RECORDS STORED SO FAR IN SV
3710  IF  PN>SV  THEN  PRINT"YOU  ONLY  HAVE";A$;"   RECORDS
SAVED":FOR
     I=1TO2000:NEXT I:GOTO3650
3720   IF   PN>200   THEN  PRINT"MAX  NUMBER  IS  200":  FOR
I=1TO2000:NEXT
     I:GOTO3650
3730 REC=PN:GOSUB 1100:REM SAVE RECORD IN SV$
3740 SV$=A$
3750 A$="INSERTING BEFORE WORD #"
3760 GOSUB 2430:REM GET NEW DATA WORD AND SAVE IT
3770 IF A$="T" THEN 3750
3780 IF A$="R" THEN RUN
3790 CLS:PRINT"INSERTING BEFORE DATA WORD #";PN
3800 FOR I=SV TO PN+1 STEP -1
3810 REC=I:GOSUB 1100:REM GET TOP RECORD IN A$
3820 REC=I+1:GOSUB 1240:REM STORE IT ONE HIGHER
3830 NEXT I
3840 REC=PN+1:A$=SV$: GOSUB 1240:REM NOW SAVE OLD RECORD
```

```
3850 REC=201:A$=STR$(SV+1):GOSUB1240:REM  UPDATE  #  RECORDS
STORED
3860 PRINT"INSERTION COMPLETED":FOR I=1TO2000:NEXTI:RUN
```

```
10 REM ***********************************
20 REM *                                 *
30 REM *          PROGRAM DSETUP         *
35 REM *       Version 1.0, Sep 85       *
40 REM *                                 *
50 REM * THIS PROGRAM ALLOWS THE USER    *
60 REM * TO INPUT DISPLAY DATA PAGE      *
70 REM * SPECIFICATIONS INTO THE PCM     *
80 REM * MONITOR SYSTEM.  IT STORES      *
90 REM * CREATED PAGES IN THE FILE,      *
100 REM * DISPLAY.DAT.                    *
110 REM *                                *
120 REM ***********************************
130 REM
140 REM ***************************
150 REM *                         *
160 REM * VARIABLE DEFINITIONS    *
170 REM *                         *
180 REM ***************************
190 REM
200 ' MR$=MAX NUMBER OF RECORDS STORED IN DISPLAY.DAT
210 ' NAME$=NAME OF DATA FILE TO ACCESS
220 ' SN() = SYNC NUMBER. A CONSTANT FOR BIN-DEC CONVERSION
230 ' CORRECTION = CORRECTION FACTOR TO OFFSET PCM MONITOR
    DELAYS
240 '
250 '
260 ' THESE ADDRESSES ARE USED FOR INFORMATION  TRANSFER  TO
THE
270 ' PCM MONITOR
280  '  VLID = VALID ADDRESS TO TELL PCM PROCESSOR THAT DATA
IS
    GOOD
290 ' BITRATE = ADDRESS TO STORE BIT RATE IN
300 ' PCMTYPE = ADDRESS TO STORE PCM TYPE, NRZ OR BI-PHASE
310 ' WORDLENGTH = ADDRESS FOR BITS PER PCM WORD
320 ' FPERBUS = ADDRESS FOR # OF FRAMES PER BUFFER MEMORY.
330 ' FRAMELENGTH = ADDRESS FOR NUMBER OF WORDS PER FRAME
340 ' SH,SM,SL = SYNC WORD ADDRESSES (HIGH, MIDDLE, LOW)
350 ' EROR = ERROR ACCUMULATOR
360 ' PLARITY = ADDRESS FOR POLARITY, NORMAL OR INVERSE
370 ' CALRATE = ADDRESS SHOWING WHAT THE CALCULATED PCM RATE
IS
380 '
390 '
400 '
410 ' REC = RECORD NUMBER IN NAME$ TO ACCESS
420 ' B$=BITS PER WORD
430 ' B=MAX VALUE OF WORD ACCORDING TO B$
440 ' W$=WORDS PER FRAME
450 ' TY$=TYPE OF PCM: NRZ-L OR BI-PHASE-L
460 ' TY=VARIBLE TO SHOW WHAT PCM TYPE IS, 1=NRZ, 2=BI-PHASE
```

```
470 ' RA$=PCM RATE: AUTO OR VALUE
480 ' RA=VARIABLE TO SHOW BIT RATE, AUTO OR NUMBER
490 ' O$=PCM ORDER: MSB FIRST OR LSB FIRST
500 ' O=VARIABLE TO SHOW WHAT  PCM  RDER  IS,  1=MSB,  2=LSB
FIRST
510 ' PO$=PCM POLARITY: NORMAL OR INVERSE
520 ' PO=VARIABLE TO SHOW WHAT POLARITY PCM IS. 1=NORMAL,
     2=INVERSE
530 ' SW$=# SYNC WORDS
540 ' SW$(1-3)=VALUE OF SYNC WORD 1-3
550 ' SW(1-3)=DECIMAL ACCUMULATOR FOR SYNC WORD VALUE
560 ' FB=NUMBER OF FRAMES PER BUFFER
570 ' DA=DATA ADDRESS FOR DATA WORD IN BUFFER
580 ' BO=BIT OFFSET USED IN BIN-DEC CONVERSION OF SYNC WORDS
590 ' I, J = COUNTER VARIABLES
600 ' X=GENERAL PURPOSE RESULT VARIABLE
610 ' TIME=CURSOR FLASH RATE
620 ' SP(I)=SCREEN POSITION FOR DISPLAY LINE I
630 ' VP(I)=SCREEN POSITION FOR NON-BINARY DATA IN LINE I
640 ' BP(I)=SCREEN POSITION FOR BINARY DATA IN LINE I
650 ' TP=TYPE OF PAGE. 1=MENU, 2=ENG DATA
660 ' NC=NUMBER OF CHARACTERS ON LINE USED IN DISPLAY PAGE
     ROUTINE
670 ' SAME=USED IN GET ROUTINE TO TELL IF KEY WAS PRESSED
680 ' PSN=POSITION POINTER USED IN GET ROUTINE
690 ' CH=CHARACTER READ FROM SCREEN IN GET ROUTINE
700 ' KY=VALUE OF KEY USED IN GET ROUTINE
710     '     SIZE=SIZE   OF   DATA   RECORD.     PARAM.DAT=60,
DISPLAY.DAT=250
720 ' P$=PARAMETER NUMBER
730 ' U$=PARAMETER UNITS
740 ' MN()=MINIMUM VALUE FOR PARAMETER
750 ' MX()=MAXIMUM VALUE OF PARAMETER
760 ' AL()=ALARM LOW VALUE
770 ' AH()=ALARM HIGH VALUE
780 ' D$()=DISPLAY TYPE, DECIMAL, HEX ETC
790 ' W$()=#WORDS=FRAMELENGTH
800 ' FC=FRAME COUNTER
810 ' SCALE()=SCALE FACTOR FOR DATA MAPPING  INTO  PARAMETER
RANGE
820 ' LN=SCREEN LINE NUMBER USED IN GET ROUTINE
830 ' N=NUMBER OF CHARACTERS USED IN GET ROUTINE
840 ' LP=LINE POINTER, POINTS TO DISPLAY TYPE IN MAIN LOOP
850 ' TP=TYPE POINTER, POINTS TO DISPLAY TYPE
860 '
870 REM START OF PROGRAM DSETUP
880 CLEAR 2000
890 GOTO3710:REM START OF PROGRAM
900 REM ************************
910 REM *                      *
920 REM *     INITIALIZATION   *
930 REM *        ROUTINE       *
```

```
940 REM *                          *
950 REM **************************
960 REM
970 CLS
980 POKE 65495,0:REM SET FAST SPEED
990 DIM SP(16)
1000 FOR I=1TO10:READ SP(I):NEXTI
1010 DATA 134,166,198,230,262,294,326,358,390,422
1020 DT$(1)="DEC": DT$(2)="HEX": DT$(3)="OCT": DT$(4)="BIN":
     DT$(5)="SWI"
1030 RETURN
1040 REM
1050 REM  ************************
1060 REM  *                      *
1070 REM  * LINE INPUT ROUTINE *
1080 REM: * SP= SCREEN POSITION *
1090 REM: * RETURNS STRING IN A$*
1100 REM: * N = MAX CHARACTERS  *
1110 REM: * I = TEMP COUNTER    *
1120 REM: * PSN = LOCAL POSITION*
1130 REM  * LN = LINE OF ENTRY  *
1140 REM: *                      *
1150 REM: ************************
1160 REM
1170 SAME=0:PSN=1:PRINT@SP(LN),"["+A$;:PRINT@SP(LN)+N+1,"]";
1180 REM MAKE FLASHING CURSOR
1190 IF PSN=N+1 THEN 1240:REM END OF LINE
1200 X=SP(LN)+PSN+1024:REM DIRECT READ OF SCREEN
1210 CH=PEEK(X)
1220    POKE    X,109:FOR   I=1TO40:NEXT   I:POKE   X,CH:FOR
I=1TO20:NEXT I
1230 REM 109 = "-"
1240 A$=INKEY$: IF A$="" THEN 1190
1250 KY=ASC(A$)
1260 IF KY=12 THEN QUIT=1:GOTO 1350:REM CLEAR
1270 IF KY=13 THEN 1350:REM ENTER
1280 IF KY=8 AND PSN=1 THEN 1240:REM LEFT ARROW
1290 IF KY=8 THEN PSN=PSN-1:GOTO1240
1300 IF KY=10 OR KY=94 THEN 1360:REM USE UP AND DOWN ARROWS
1310 IF PSN <1 OR PSN >N THEN  1240  ELSE  PRINT@SP(LN)+PSN,
A$;
1320 IF KY=9 THEN 1330 ELSE SAME=1:REM NEW DATA ENTERED
1330 PSN=PSN+1
1340 GOTO1240
1350 REM COLLECT CHARACTERS INTO A$
1360 A$=""
1370  FOR  PSN=1  TO  N:X=PEEK(SP(LN)+PSN+1024):IF X>95 THEN
X=X-64
1380 A$=A$+CHR$(X):NEXT PSN
1390 IF VAL(A$)>200 THEN 1170:REM MAX NUMBER OF RECORDS
1400  PRINT@SP(LN),"  ";:PRINT@SP(LN)+N+1,"  ";:REM   REMOVE
BRACKETS
```

```
1410 RETURN
1420 REM
1430 REM
1440 REM ************************
1450 REM *                      *
1460 REM *  DISK INPUT ROUTINE  *
1470 REM *                      *
1480 REM *  RECORD # IN PN      *
1490 REM *  RECORD NAME IN NAME$*
1500 REM *  RETURNS DATA IN A$  *
1510 REM *                      *
1520 REM ************************
1530 IF REC=0 THEN RETURN
1540 OPEN"D", #1, NAME$,SIZE
1550 GET#1, REC
1560 INPUT #1,A$
1570 CLOSE #1
1580 RETURN
1590 REM ************************
1600 REM *                      *
1610 REM *  DISK OUTPUT ROUTINE *
1620 REM *                      *
1630 REM *  RECORD # IN PN      *
1640 REM *  RECORD NAME IN NAME$*
1650 REM *  DATA IN A$          *
1660 REM *                      *
1670 REM ************************
1680 IF REC=0 THEN RETURN
1690 OPEN "D", #1, NAME$, SIZE
1700 WRITE #1,A$
1710 PUT #1,REC
1720 CLOSE #1
1730 RETURN
1740 REM
1750 REM
1760 REM
1770 REM ************************
1780 REM *                      *
1790 REM * LIST DATA WORD NAMES*
1800 REM *     SUBROUTINE       *
1810 REM *                      *
1820 REM ************************
1830 GOTO 1910
1840 CLS:LN=4:PRINT"  CATALOG LISTING OF";A1$;" WORDS";
1850 PRINT@32,STRING$(32,"*");
1860 PRINT"      WORD  #       NAME  PARM"
1870 PRINT STRING$(32,"*");
1880 PRINT@448,STRING$(32,"*");
1890 PRINT@128,"";
1900 RETURN
1910 REC=201:NAME$="PARAM.DAT":SIZE=60:GOSUB 1530
1920 A1$=A$:REM SAVE NUMBER OF REC
```

```
1930 GOSUB 1840:REM PRINT PAGE
1940 FOR I=1 TO VAL(A1$)
1950 REC = I
1960 GOSUB 1530:REM READ IN RECORD
1970 PRINT@LN*32,"       ";I,LEFT$(A$,7);" ";MID$(A$,8,3);
1980 LN=LN+1
1990 IF INT(I/10)=I/10 THEN 2010
2000 NEXT I
2010 PRINT@480," CR=CONT, E=EXIT ";:INPUT A$
2020 IF A$="E" THEN 2060
2030 IF I=>VAL(A1$) THEN 2060
2040 GOSUB 1840
2050 NEXT I
2060 RETURN
2070 '
2080 REM ************************
2090 REM *                      *
2100 REM *  OUTPUT DISPLA PAGE  *
2110 REM *       SUBROUTINE     *
2120 REM *                      *
2130 REM ************************
2140 CLS:PRINT A2$;PN
2150 PRINT"*********************************";
2160 PRINT"  POS  DW#  PAR   NAME    DISP"
2170 PRINT"---------------------------------";
2180   FOR   I=1TO9:PRINT"    ";I;"   ";DW$(I);"   ";P$(I);"
";N$(I);"
    ";D$(I):NEXT I
2190  PRINT"   10    ";DW$(10);"    ";P$(10);"    ";N$(10);"
";D$(10)
2200 PRINT"*********************************";
2210 PRINT" ENTER=NEXT CLEAR=ACCEPT SP=TOG";
2220 RETURN
2230 REM
2240 REM
2250 REM
2260 REM ************************
2270 REM *                      *
2280 REM * LIST DISPLAY CATALOG*
2290 REM *       SUBROUTINE     *
2300 REM *                      *
2310 REM ************************
2320 GOTO 2400:REM ENTRY POINT FOR SUBROUTINE
2330 CLS:PRINT"  CATALOG LISTING OF";A1$;" PAGES"
2340 PRINT STRING$(32,"*");
2350 PRINT"PG!1!!2!!3!!4!!5!!6!!7!!8!!9!!10";
2360 PRINT STRING$(32,"*");
2370 PRINT@448,STRING$(32,"*");
2380 PRINT@128,"":REM SET TO TOP OF DISPLAY AREA
2390 RETURN
2400 CLS:J=1:LN=4:REM LINE COUNTERS
2410 NAME$="DISPLAY.DAT":REC=101:SIZE=250:GOSUB 1530:REM GET
```

```
#
     DISPLAYS IN FILE
2420 A1$=A$:REM SAVE # PAGES
2430 GOSUB 2330:REM PRINT DISPLAY (INTERNAL SUBROUTINE)
2440  FOR  I=1  TO  2*VAL(A1$)  STEP  2:REM  2 RECORDS FOR 1
DISPLAY
     PAGE
2450 REC=I
2460 GOSUB 1530:REM READ IN FIRST RECORD
2470 REM GET LINE TO PRINT IN LI$
2480 LI$=LEFT$(A$,3) + MID$(A$,44,3) + MID$(A$,87,3) +
     MID$(A$,130,3) + MID$(A$,173,3)
2490 REC=I+1:GOSUB 1530:REM NOW GET SECOND HALF
2500 LI$=LI$ + LEFT$(A$,3) + MID$(A$,44,3) + MID$(A$,87,3) +
     MID$(A$,130,3) + MID$(A$,173,3)
2510 PRINT@LN*32,RIGHT$(STR$(J),2):PRINT@LN*32+2,LI$;
2520 IF INT(J/10)=J/10 THEN LN=4:GOTO 2550
2530 J=J+1:LN=LN+1:REM INCREMENT LINE
2540 NEXT I
2550 PRINT@480,"NUMB=SEL, ENTER=CONT, E=EXIT";:INPUT A$
2560 IF A$="E" THEN 2620
2570 IF VAL(A$)<>0 THEN 2620
2580 IF J=>VAL(A1$) THEN A$="E":GOTO 2620
2590 GOSUB 2330:REM RE-DRAW DISPLAY
2600 J=J+1:REM NOW GET NEXT RECORD
2610 NEXT I
2620 RETURN
2630 REM
2640 REM
2650 REM ************************
2660 REM *                      *
2670 REM *    RETRIEVE DATA     *
2680 REM *     SUBROUTINE       *
2690 REM *                      *
2700 REM ************************
2710 GOSUB2140:REM OUTPUT DISPLAY PAGE
2720 QUIT=0
2730 LN=1:N=3:A$=""
2740 GOSUB 1170:REM GET DATA WORD NUMBER IN A$
2750 IF SAME=0 THEN 2850:REM SAME DATA SO DON'T READ AGAIN
2760 IF VAL(A$)=0 THEN 2740:REM NOT A NUMBER
2770 DW$(LN)=A$:REM SAVE DATA WORD NUMBER
2780 REM NOW READ IN PARAMETER VALUES
2790  REC=VAL(A$):NAME$="PARAM.DAT":SIZE=60:GOSUB   1530:REM
GET
     PARAMETER
2800 TP$(LN)=A$:REM SAVE TOTAL PARAMETER VALUES OR CHANGES
     THERETO
2810 N$(LN)=LEFT$(A$,7):P$(LN)=MID$(A$,8,3)
2820 PRINT@(LN+3)*32+12,P$(LN);:PRINT"  ";N$(LN);
2830 GOSUB3570:REM GET DISPLAY TYPE IN DT$
2840 D$(LN)=DT$(J):REM SAVE IT IN D$
```

```
2850 IF QUIT=1 THEN 2910:REM CLEAR KEY PRESSED
2860 IF KY=94 THEN LN=LN-2:REM BACKUP
2870 A$=""
2880 LN=LN+1:IF LN=11 THEN GOTO 2730
2890 IF LN=0 THEN LN=10
2900 GOTO 2740
2910 CLS:PRINT:PRINT:PRINT"S)AVE DISPLAY #";PN
2920 PRINT"T)RY AGAIN?"
2930 PRINT"R)ETURN TO MENU?"
2940 PRINT:INPUT"WHICH ONE>";A$
2950 IF A$="S" THEN 2990
2960 IF A$="T" THEN 2710
2970 IF A$="R" THEN RUN:REM START OVER WITH MAIN MENU
2980 GOTO 2910
2990 A$="":FOR I=1TO5:REM COLLECT 5 LINES
3000 D$(I)=LEFT$(D$(I),1):REM SAVE ONLY 1 CHAR
3010          IF          TP$(I)=""          THEN          TP$(I)="
":REM FILL NULL STRING
3020 A$=A$+DW$(I)+TP$(I)+D$(I):REM  COLLECT  FIRST  HALF  OF
DISPLAY
     DATA
3030 NEXT I
3040 NAME$="DISPLAY.DAT":SIZE=250
3050 REC=2*PN-1:REM SAVE FIRST PART OF PAGE
3060 GOSUB 1680
3070 A$="":FOR I=6 TO 10
3080 D$(I)=LEFT$(D$(I),1)
3090          IF          TP$(I)=""          THEN          TP$(I)="
"
3100 A$=A$+DW$(I)+TP$(I)+D$(I):NEXT I
3110 REC=2*PN:GOSUB 1680
3120 PRINT:PRINT"DATA WORD #";PN;N$;" SAVED"
3130 FOR I=1TO2000:NEXT I
3140 A$="S":REM SHOW CALLING PROGRAM THAT REC WAS SAVED
3150 RETURN
3160 REM
3170 REM ************************
3180 REM *                      *
3190 REM *  DISPLAY DATA ENTRY  *
3200 REM *    PAGE SUBROUTINE   *
3210 REM *                      *
3220 REM ************************
3230 REM
3240 CLS:PRINT@484,"PRESS L TO LIST, E TO EXIT
     ":PRINT@35,A2$;:INPUT A$
3250 IF A$="L" THEN GOSUB 2320:IF A$="E" THEN 3240
3260 IF A$="E" THEN RETURN:REM DON'T EDIT
3270 PN=VAL(A$): IF PN=0 THEN 3240:REM MUST BE NUMBER
3280 NAME$="DISPLAY.DAT":SIZE=250:REC=101:GOSUB 1530
3290 MR$=A$:REM SAVE MAX RECORDS
3300 IF PN>VAL(A$) THEN PRINT"YOU ONLY HAVE ";A$;" RECORDS":
     FOR I=1TO2000:NEXT I:GOTO3240
```

```
3310 IF PN>50 THEN PRINT"MAX NUMBER IS 50":FOR I=1TO2000:
     NEXT I:GOTO3240
3320 REC=2*PN-1:REM GET FIRST HALF
3330 GOSUB 1530:REM GET OLD DATA
3340 FOR I=1 TO 10
3350 IF I<6 THEN X=(I-1)*43+1 ELSE X=(I-6)*43+1
3360 IF I=6 THEN REC=2*PN:GOSUB 1530:REM GET  NEXT  HALF  OF
RECORD
3370 TP$(I)=MID$(A$,X+3,39):REM SAVE DATA WORD IN TP$ AS IT
     COMES IN
3380 DW$(I)=MID$(A$,X,3)
3390 N$(I)=MID$(A$,X+3,7)
3400 P$(I)=MID$(A$,(X+10),3)
3410 D$(I)=MID$(A$,X+42,1)
3420 IF D$(I)="D" THEN D$(I)="DEC"
3430 IF D$(I)="H" THEN D$(I)="HEX"
3440 IF D$(I)="O" THEN D$(I)="OCT"
3450 IF D$(I)="B" THEN D$(I)="BIN"
3460 IF D$(I)="S" THEN D$(I)="SWI"
3470 NEXT I
3480 RETURN
3490 REM ************************
3500 REM *                      *
3510 REM *    GET DATA TYPE     *
3520 REM *     SUBROUTINE       *
3530 REM *                      *
3540 REM ************************
3550 REM
3560 REM
3570 J=1
3580 PRINT@(LN+3)*32+26,"    ";
3590 FOR I=1TO40:NEXT I:PRINT@(LN+3)*32+26,DT$(J);:
     FOR I=1TO40:NEXT I
3600 KY$=INKEY$:IF KY$="" THEN 3580
3610 IF ASC(KY$)=13 THEN RETURN:REM ENTER
3620 IF ASC(KY$)=12 THEN QUIT=1:RETURN:REM CLEAR
3630 J=J+1:IF J>5 THEN J=1
3640 GOTO 3580
3650 REM
3660 REM ************************
3670 REM *                      *
3680 REM *    START OF PROGRAM  *
3690 REM *                      *
3700 REM ************************
3710 GOSUB 970:REM INIT
3720 CLS
3730 PRINT@0, STRING$(32,"*");
3740 PRINT"*   DISPLAY DEFINITION PAGE    *";
3750 PRINT STRING$(32,"*")
3760 PRINT@128,"A)DD NEW DISPLAY PAGE"
3770 PRINT"E)DIT DISPLAY PAGE"
3780 PRINT"I)NSERT DISPLAY PAGE"
```

```
3790 PRINT"D)ELETE DISPLAY PAGE"
3800 PRINT"L)IST CURRENT DISPLAY PAGES"
3810 PRINT"P)ARAMETER LISTING"
3820 PRINT"Q)UIT"
3830 PRINT:INPUT"WHICH ONE";KY$
3840 IF KY$="A" THEN 3970
3850 IF KY$="Q" THEN RUN"MAIN"
3860 IF KY$="E" THEN 4140
3870 IF KY$="D" THEN 4250
3880 IF KY$="I" THEN 4430
3890 IF KY$="L" THEN GOSUB 2400:REM LIST DISPLAY PAGES
3900  IF  KY$="P"  THEN  GOSUB  1830:REM  LIST PARAMETERS IN
RECORD
3910 GOTO 3720:REM WRONG KEY? THEN START AGAIN
3920 REM ************************
3930 REM *                      *
3940 REM *   ADD DISPLAY PAGE   *
3950 REM *                      *
3960 REM ************************
3970 NAMES$="DISPLAY.DAT":REC=101:SIZE=250:REM MAX # RECORDS
     COUNT
3980 GOSUB 1530:REM READ # RECORDS IN FILE
3990 PN=VAL(A$)+1:REM MAKE A NEW RECORD NUMBER
4000 A2$="    ADDING DISPLAY PAGE #"
4010 GOSUB 2710:REM GET NEW DATA
4020 IF A$="T" THEN 4010:REM TRY AGAIN
4030 PRINT:PRINT"UPDATING TOTAL # OF RECORDS"
4040 NAMES$="DISPLAY.DAT":REC=101:A$=STR$(PN)
4050 GOSUB 1680:REM WRITE NEW MAX RECORDS
4060 RUN: REM START AGAIN WITH  MAIN  MENU.   CLEARS  STRING
SPACE
4070 REM
4080 REM
4090 REM ************************
4100 REM *                      *
4110 REM *     EDIT DISPLAY     *
4120 REM *                      *
4130 REM ************************
4140 A2$="ENTER DISPLAY # TO EDIT"
4150 GOSUB 3240:REM GET DISPLAY INFO
4160 A2$="       EDITING DISPLAY #"
4170 GOSUB 2710:REM NOW EDIT IT
4180 IF A$="T" THEN 4160
4190 RUN: REM USE RUN TO CLEAR STRING SPACE
4200 REM ************************
4210 REM *                      *
4220 REM * DELETE DISPLAY PAGE *
4230 REM *                      *
4240 REM ************************
4250 A2$="   DELETE DISPLAY # "
4260 GOSUB 3240:REM READ DATA
4270 IF A$="E" THEN RUN
```

```
4280 A2$="        DELETE DISPLAY #"
4290 GOSUB 2140:REM OUTPUT DISPLAY PAGE
4300 PRINT@480,"DELETE THIS DATA WORD? Y/N  ";:INPUT A$
4310 IF A$<>"Y" THEN 4250
4320 CLS:PRINT"      DELETING DATA WORD";PN
4330 FOR I=PN+1 TO VAL(MR$):REM MR$ HAS MAX# RECORDS FROM
     PREVIOUS CALL
4340 PRINT"      MOVING RECORD #";I
4350 REC=2*I-1:GOSUB 1530:REM READ OLD RECORD
4360 REC=REC-2:GOSUB 1690:REM PUT IN NEXT LOWER POSITION
4370 REC=2*I:GOSUB 1530:REM READ RECORD
4380 REC=REC-2:GOSUB 1690:REM WRITE IT TO PREVIOUS NUMBER
4390 NEXT I
4400 REC=101:A$=STR$(VAL(MR$)-1):REM UPDATE # RECORDS STORED
4410 GOSUB 1690
4420 RUN
4430 REM
4440 REM ************************
4450 REM *                      *
4460 REM *    INSERT DATA WORD  *
4470 REM *                      *
4480 REM ************************
4490 REM
4500   CLS:PRINT@484,"PRESS   L   TO   LIST,   E   TO   EXIT
":PRINT@32,"ENTER
     WORD# TO BE INSERTED";:INPUT A$
4510 IF A$="L" THEN GOSUB 2400:IF A$="E" THEN 4500
4520 IF A$="E" THEN RUN:REM EXIT
4530 PN=VAL(A$): IF PN=0 THEN  4500
4540 NAME$="DISPLAY.DAT":REC=101:SIZE=250:GOSUB1530
4550 MR$=A$:REM SAVE TOTAL RECORDS STORED
4560 IF VAL(MR$)=50 THEN PRINT"YOU HAVE 50 RECORDS, MUST
     DELETE";:FOR I=1TO2000:NEXT I:GOTO4500
4570 IF PN>50 THEN PRINT"MAX NUMBER IS 50": FOR I=1TO2000:
     NEXT I:GOTO4500
4580 REC=2*PN-1:GOSUB 1530:REM SAVE RECORD
4590 S1$=A$:REM SAVE FIRST HALF
4600 REC=2*PN:GOSUB 1530:REM NOW SAVE SECOND HALF
4610 S2$=A$
4620 A2$="   INSERTING BEFORE WORD #"
4630 GOSUB 2710:REM GET NEW DISPLAY
4640 IF A$="T" THEN 4500:REM START ALL OVER AGAIN
4650 IF A$="R" THEN RUN
4660 CLS:PRINT"INSERTING BEFORE DISPLAY #";PN
4670 PRINT
4680 FOR I=VAL(MR$) TO PN+1 STEP -1
4690 PRINT"   MOVING DISPLAY PAGE #";I
4700 REC=2*I:GOSUB 1530:REM GET TOP RECORD IN A$
4710 REC=REC+2:GOSUB 1680:REM STORE IT ONE HIGHER
4720 REC=2*I-1:GOSUB 1530
4730 REC=REC+2:GOSUB 1680
4740 NEXT I
```

```
4750 REC=2*PN+1:A$=S1$:GOSUB 1680:REM NOW SAVE OLD RECORD
4760 REC=REC+1:A$=S2$:GOSUB 1680
4770  REC=101:A$=STR$(VAL(MR$)+1):GOSUB  1680:REM  UPDATE  #
RECORDS
     STORED
4780 PRINT"INSERTION COMPLETED":FOR I=1TO2000:NEXTI:RUN
4790 END_
```

```
10 REM ************************************
20 REM *                                  *
30 REM *        PROGRAM EXECUTE           *
35 REM *      Version 1.0, Sep 85         *
40 REM *                                  *
50 REM * THIS PROGRAM READS DISPLAY       *
60 REM * PAGES FROM DISPLAY.DAT AND       *
70 REM * PRESENTS THEM TO THE USER        *
80 REM * IN ONE OF THREE FASHIONS.        *
90 REM *  1.  ENGINEERING UNITS           *
100 REM * 2.  BAR GRAPH REPRESENTATION     *
110 REM * 3.  PLOTS OF A SINGLE WORD       *
120 REM *                                  *
130 REM ************************************
140 ' MR$=MAX NUMBER OF RECORDS STORED IN DISPLAY.DAT
150 ' NAME$=NAME OF DATA FILE TO ACCESS
160 ' SN() = SYNC NUMBER. A CONSTANT FOR BIN-DEC CONVERSION
170 '
180 ' THESE ADDRESSES ARE USED FOR INFORMATION  TRANSFER  TO
THE
190 ' PCM MONITOR
200  '  VLID = VALID ADDRESS TO TELL PCM PROCESSOR THAT DATA
IS
     GOOD
210 ' BITRATE = ADDRESS TO STORE BIT RATE IN
220 ' PCMTYPE = ADDRESS TO STORE PCM TYPE, NRZ OR BI-PHASE
230 ' WORDLENGTH = ADDRESS FOR BITS PER PCM WORD
240 ' FPERBUF = ADDRESS FOR # OF FRAMES PER BUFFER MEMORY.
250 ' FRAMELENGTH = ADDRESS FOR NUMBER OF WORDS PER FRAME
260 ' SH,SM,SL = SYNC WORD ADDRESSES (HIGH, MIDDLE, LOW)
270 ' EROR = ERROR ACCUMULATOR
280 ' PLARITY = ADDRESS FOR POLARITY, NORMAL OR INVERSE
290 ' CALRATE = ADDRESS SHOWING WHAT THE CALCULATED PCM RATE
IS
300 ' CORRECTION = RETURNED VALUE FOR CORRECTION NEEDED  FOR
SYNC
310 ' STATUS = ADDRESS FOR RETURN CODE FOR IN SYNC OR NOT
320 ' REC = RECORD NUMBER IN NAME$ TO ACCESS
330 ' TY$=TYPE OF PCM: NRZ-L OR BI-PHASE-L
340 ' TY=VARIBLE TO SHOW WHAT PCM TYPE IS, 1=NRZ, 2=BI-PHASE
350 ' RA$=PCM RATE: AUTO OR VALUE
360 ' RA=VARIABLE TO SHOW BIT RATE, AUTO OR NUMBER
370 ' O$=PCM ORDER: MSB FIRST OR LSB FIRST
380  '  O=VARIABLE  TO  SHOW  WHAT PCM RDER IS, 1=MSB, 2=LSB
FIRST
390 ' PO$=PCM POLARITY: NORMAL OR INVERSE
400 ' PO=VARIABLE TO SHOW WHAT POLARITY PCM IS. 1=NORMAL,
     2=INVERSE
410 ' SW$=# SYNC WORDS
420 ' SW$(1-3)=VALUE OF SYNC WORD 1-3
430 ' SW(1-3)=DECIMAL ACCUMULATOR FOR SYNC WORD VALUE
440 ' FB=NUMBER OF FRAMES PER BUFFER
```

```
450 ' DA=DATA ADDRESS FOR DATA WORD IN BUFFER
460 ' DB=ALTERNATE FOR DA, POINTS TO BEFINNGING OF BUFFER
470 ' DC=ALTERNATE FOR DA, END OF BUFFER USUALLY
480 ' BO=BIT OFFSET USED IN BIN-DEC CONVERSION OF SYNC WORDS
490 ' I, J, K = COUNTER VARIABLES
500 ' X=GENERAL PURPOSE RESULT VARIABLE
510 ' TIME=CURSOR FLASH RATE
520 ' SP(I)=SCREEN POSITION FOR DISPLAY LINE I
530 ' VP(I)=SCREEN POSITION FOR NON-BINARY DATA IN LINE I
540 ' BP(I)=SCREEN POSITION FOR BINARY DATA IN LINE I
550 ' TP=TYPE OF PAGE. 1=MENU, 2=ENG DATA
560 ' NC=NUMBER OF CHARACTERS ON LINE USED IN DISPLAY PAGE
    ROUTINE
570 ' SAME=USED IN GET ROUTINE TO TELL IF KEY WAS PRESSED
580 ' PSN=POSITION POINTER USED IN GET ROUTINE
590 ' CH=CHARACTER READ FROM SCREEN IN GET ROUTINE
600 ' KY=VALUE OF KEY USED IN GET ROUTINE
610 ' KY$=STRING OF KEY INPUT
620   '   SIZE=SIZE   OF    DATA    RECORD.    PARAM.DAT=60,
DISPLAY.DAT=250
630 ' P$=PARAMETER NUMBER
640 ' U$=PARAMETER UNITS
650 ' MN()=MINIMUM VALUE FOR PARAMETER
660 ' MX()=MAXIMUM VALUE OF PARAMETER
670 ' AL()=ALARM LOW VALUE
680 ' AH()=ALARM HIGH VALUE
690 ' D$()=DISPLAY TYPE, DECIMAL, HEX ETC
700 ' W$()=#WORDS=FRAMELENGTH
710 ' B$=STRING CONTANING BITS PER WORD
720 ' B=VALUE OF B$
730 ' FC=FRAME COUNTER
740 ' FD=ALTERNATE FRAME COUNTER
750  '  SCALE()=SCALE FACTOR FOR DATA MAPPING INTO PARAMETER
RANGE
760 ' LN=SCREEN LINE NUMBER USED IN GET ROUTINE
770 ' N=NUMBER OF CHARACTERS USED IN GET ROUTINE
780 ' LP=LINE POINTER, POINTS TO DISPLAY TYPE IN MAIN LOOP
790 ' TP=TYPE POINTER, POINTS TO DISPLAY TYPE
800 ' LM=LINE NUMBER OF SELECTED DISPLAY TYPE  IN  ENG  DISP
PAGE
810 ' BUFFER=START OF MEMORY BUFFER ($E000)
820 CLEAR 1000: REM MAKE ROOM FOR STRINGS
830  TEST=0:REM  SET  TEST MODE SO SOFTWARE WILL RUN WITHOUT
PCM        PROCESSOR
840 GOTO 5620:REM START OF PROGRAM
850 REM ***************************
860 REM *                         *
870 REM *      INITIALIZATION     *
880 REM *         ROUTINE         *
890 REM *                         *
900 REM ***************************
910 REM
```

```
920 POKE 65495,0:REM SET FAST SPEED
930 CLS
940 '
950 '
960 '
970 '
980 ' SET UP FUNCTIONS FOR HIGH AND LO BYTE POKES
990 DEF FNBL(X)=X-256*INT(X/256)
1000 DEF FNBH(X)=INT(X/256)
1010 '
1020 '
1030 '
1040 '
1050 '*******************************
1060 '*                             *
1070 '*      I/O CONTROL BLOCK      *
1080 '*          EQUATES            *
1090 '*                             *
1100 '*******************************
1110 BUFFER=57344:REM $E000 = START OF DATA INPUT BUFFER
1120 VLID = 59390:REM $E7FE
1125 POKE VLID,255:REM STOP SYSTEM IF RUNNING
1130 BITRATE = 59388:REM $E7FC 0=CAL, NUMBER=RATE
1140 PCMTYPE = 59386:REM $E7FA, C,N,B
1150 WORDLENGTH = 59384:REM $E7F8, 8-16
1160 FRAMELENGTH = 59382:REM $E7F6
1170 SH=59380:REM $E7F4  FRAME SYNC HIGH
1180 SM=59378:REM $E7F2 FRAME SYNC MIDDLE
1190 SL=59376:REM $E7F0 FRAME SYNC LOW
1200 EROR=59374:REM $E7EE
1210 PLARITY = 59372:REM $E7EC NORMAL R INVERSE
1220 FPERBUF = 59370:REM $E7EA
1230 RPCMTYPE = 59368:REM $E7E8 RETURNED PCM TYPE
1240 CALRATE = 59366:REM $E7E6
1250 CORRECTION = 59364:REM $E7E4
1260 STATUS = 59362:REM $E7E2 0 IF NOT IN SYNC, 1 OR MORE IF
IN        SYNC
1270 WC = 59360:REM $E7E0:  WORD  COUNTER  OR  DATA  ADDRESS
RETURNED       FOR SYNCHRONIZING TRACE MODE
1280 '
1290 '
1300 '
1310 '
1320 ' READ IN NUMBER OF RECORDS STORED
1330 NAME$="DISPLAY.DAT":SIZE=250:REC=101:GOSUB4580
1340 MR$=A$
1350 '
1360 '
1370 '
1380 '
1390 REM READ IN FRAME SPECIFICATIONS
1400 DIM SN(16)
```

```
1410 DIM Y(30):REM USED IN PLOT ROUTINE
1420 DIM N$(11):REM NAME OF VARIABLE
1430 DIM D$(11):REM DATA TYPE OF VARIABLE
1440  SN(1)=32768:  SN(2)=16384:   SN(3)=8192:   SN(4)=4096:
SN(5)=2048: SN(6)=1024: SN(7)=512:SN(8)=256:
     SN(9)=128:SN(10)=64
1450 SN(11)=32:SN(12)=16:SN(13)=8:SN(14)=4:SN(15)=2:SN(16)=1
1460 REC=102:GOSUB4580
1470 '
1480 '
1490 '
1500 ' CONVERT A$ TO FRAME VALUES
1510 '
1520   B$=LEFT$(A$,2):   W$=MID$(A$,3,3):  TY$=MID$(A$,6,8):
RA$=MID$(A$,14,4): O$=MID$(A$,18,3):
     PO$=MID$(A$,21,7)
1530        SW$=MID$(A$,28,1):        SW$(1)=MID$(A$,29,16):
SW$(2)=MID$(A$,45,16): SW$(3)=MID$(A$,61,16)
1540 PRINT:PRINT:PRINT"  PLEASE SET ORDER SWITCH TO ";O$
1550 DEFUSR1=4096:REM SET TO $1000
1560 LOADM"CONVERT":REM MACHINE PROGRAM TO DO CONVERSIONS
1570 TY=1:IF TY$="NRZ      " THEN TY=2
1580 IF TY$="BI-PHASE" THEN TY=3
1590 IF RA$<>"auto" THEN RA=2 ELSE RA=1
1600 IF O$="MSB" THEN O=2 ELSE O=1
1610 IF PO$="INVERSE" THEN PO=2 ELSE PO=1
1611 TIME=50:REM CURSOR FLASH RATE
1612 LM=1:REM LINE POINTER FOR TYPE IN ENG PAGE
1613 '
1614 '
1615 ' SET UP SCREEN POSITIONS
1616 '
1617 SP(0)=27: REM DISPLAY NUMBER
1618 FOR I=1 TO 10
1619 SP(I)=32*I+96:VP(I)=SP(I)+14:YP(I)=SP(I)+1055
1620 NEXT I
1621   VH=1536:VO=1537:REM   VARIABLE  STORAGE  FOR  CONVERT
ROUTINE          ($600)
1622 RETURN
1623 '
1630 REM ************************
1631 REM *                      *
1632 REM *   SET I/O CONTROLS   *
1633 REM *     SUBROUTINE       *
1634 REM *                      *
1635 REM ************************
1650 '
1660 ' MOST VALUES ARE 8 BIT SO SET HIGH BYTE=0
1670 ' LOW ADDRESS HERE = HIGH BYTE FOR PCM PROCESSOR
1680 POKE BITRATE,0:POKE BITRATE+1,0
1690 IF RA<>2 THEN 1730:REM AUTO IS SELECTED
1700 RA=INT(16000/VAL(RA$)-CORRECTION):REM CALCULATE RATE AS
```

```
A         WORD COUNT
1710 POKE BITRATE, FNBH(RA):REM CALC HIGH BYTE
1720 POKE BITRATE+1,FNBL(RA):REM LO BYTE
1730 POKE PLARITY,0:POKE PLARITY+1,ASC(LEFT$(PO$,1))
1740 POKE PCMTYPE,0:POKE PCMTYPE+1,ASC("C"):REM C IS DEFAULT
1750 IF TY=2 THEN POKE PCMTYPE+1,ASC("N"):REM N FOR NRZ
1760 IF TY=3 THEN POKE PCMTYPE+1,ASC("B"):REM B FOR BI-PHASE
1770  FB=INT(1016/VAL(W$)):REM  MAX  NUMBER  OF  FRAMES  PER
BUFFER
1780 POKE FPERBUF,0:POKE FPERBUF+1,FB
1790         POKE        FRAMELENGTH,        FNBH(VAL(W$)):POKE
FRAMELENGTH+1,FNBL(VAL(W$))
1800    POKE    WORDLENGTH,0:POKE    WORDLENGTH+1,VAL(B$)-1:REM
OFFSET TO       BASE 0
1810 POKE EROR,0:POKE EROR+1,0
1820 POKE CALRATE,0:POKE CALRATE+1,0:REM INIT CALRATE
1830 POKE CORRECTION,0:POKE CORRECTION+1,0:REM INIT
1840 '
1850 '
1860 ' CONVERT SYNC WORD BINARY STRING TO NUMBER
1870 BO=16-VAL(B$):REM BIT OFFSET SINCE SW IS LEFT JUSTIFIED
1880 FOR J=1 TO 3
1890 FOR I=1 TO VAL(B$)
1900 IF VAL(MID$(SW$(J),I,1))=0 THEN 1920
1910 SW(J)=SW(J)+SN(I+BO)
1920 NEXT I
1930 NEXT J
1940 POKE SH, FNBH(SW(1)):POKE SH+1, FNBL(SW(1))
1950 POKE SM, FNBH(SW(2)):POKE SM+1, FNBL(SW(2))
1960 POKE SL, FNBH(SW(3)):POKE SL+1, FNBL(SW(3))
1970 POKE WC,0:POKE WC+1,0:REM INIT WORD COUNTER TO 0 WORDS
1980 '
1990 '
2000 ' NOW TELL PCM PROCESSOR THAT DATA IS VALID
2010 POKE VLID+1,0:POKE VLID,0
2020 RETURN
2030 '
2160 REM
2170 REM ************************
2180 REM *                      *
2190 REM *     HEADER OUTPUT    *
2200 REM *       SUBROUTINE     *
2210 REM ************************
2220 REM
2230 REM
2240 REM TP=TYPE PAGE. 1 FOR MENU, 2 FOR ENG DATA
2250 CLS:NC=30
2260 IF TP=2 THEN NC=32:GOTO 2290
2270 PRINT STRING$(32,"*");
2280 PRINT "*";
2290 FOR I=1 TO (NC-LEN(A$)-LEN(PG$))/2:PRINT" ";:NEXT I
2300 PRINT A$;
```

```
2310 IF POS(0)=31 THEN 2330
2320 PRINT " ";:GOTO2310
2330 IF TP=2 THEN PRINT " "; ELSE PRINT "*";
2340 PRINT STRING$(32,"*");
2350 RETURN
2360 REM ***************************
2370 REM *                         *
2380 REM *     ENGINEERING DATA    *
2390 REM * HEADER OUTPUT ROUTINE   *
2400 REM *                         *
2410 REM ***************************
2420 REM
2430 A$="ENGINEERING DISPLAY PAGE    ":TP=2
2440 GOSUB 2250
2450 PRINT"DW#    NAME     VALUE      UNITS TY";
2460 PRINT STRING$(32,"*");
2470 PRINT@448,STRING$(32,"*");
2480   PRINT  CHR$(95);CHR$(94);"  NUM,  T)YP,  B)AR,  P)LT,
E)XIT";
2490 PRINT@128,"";:REM SET TOP OF SCREEN
2500 RETURN
2510 REM ***************************
2520 REM *                         *
2530 REM *        PLOT POINT       *
2540 REM *        SUBROUTINE       *
2550 REM *                         *
2560 REM * THIS ROUTINE PLOTS A    *
2570 REM * POINT AT X,Y ABSOLUTE   *
2580 REM *                         *
2590 REM * X=0-31, Y=0-99          *
2600 REM ***************************
2610 Y=INT(Y/5)/2+3:REM CONVERT TO ABSOLUTE COORDINATES
2620 IF INT(Y)=Y THEN CH=147 ELSE CH=156
2630 Y=480-32*INT(Y)+X
2640 PRINT@Y(J),CHR$(144);:REM ERASE OLD POINT
2650 PRINT@Y,CHR$(CH);:REM PLOT NEW POINT
2660 Y(J)=Y:REM SAVE POINT ADDRESS
2670 RETURN
2680 REM ***************************
2690 REM *                         *
2700 REM *        BAR CHART       *
2710 REM *         ROUTINE         *
2720 REM *                         *
2730 REM ***************************
2740 ' FIRST DRAW CHART
2750 CLS:PRINT" BAR GRAPH DISPLAY FOR PAGE";PN
2760 PRINT STRING$(32,"*");
2770 PRINT RIGHT$("%",1):REM TRICK TO PRINT IN FIRST COLUMN
2780 FOR J=90 TO 0 STEP -10
2790 PRINT RIGHT$(STR$(J),2);CHR$(154):REM PRINT VERT LINE
2800 NEXT J
2810 FOR J=1TO10:Y(J)=0:NEXT J
```

```
2820 PRINT@416,"  ";CHR$(156);:REM LOWER LEFT CORNER
2830 '
2840 '
2850 ' NOW PRINT DATA WORD NUMBERS
2860 FOR J=1TO 10
2870 PRINT MID$(DW$(J),1,1);CHR$(156);CHR$(156);
2880 NEXT J
2890 PRINT@448,"DW ";
2900 FOR J=1TO 10
2910 PRINT MID$(DW$(J),2,1);"  ";
2920 NEXT J
2930 PRINT@480," ";
2940 FOR J=1TO10
2950 PRINT "  ";MID$(DW$(J),3,1);
2960 NEXT J
2970 '
2980 '
2990 ' PLOT ALARM LIMITS
3000 FOR J=1 TO 10
3005 IF AL(J)=0 AND AH(J)=0 THEN 3160:REM NO ALARMS
3010 L=MX(J)-MN(J):IF L=0 THEN 3160:REM MUST BE SWITCH
3020 YH=INT(20*(AH(J)-MN(J))/L)/2+3
3030 YL=INT(20*(AL(J)-MN(J))/L)/2+3
3040 X=3*J+1:REM HORIZ POS
3050 IF YH=INT(YH) THEN CH=130 ELSE CH=138:REM  BOT  LFT  OR
ALL          LFT
3060 YH=INT(YH)
3070 Y=480+X-32*YH:PRINT@Y,CHR$(CH);
3080  IF  YL=INT(YL) THEN CH=138 ELSE CH=136:REM ALL LEFT OR
BOT        LFT
3090 YL=INT(YL)
3100 Y=480+X-32*YL:PRINT@Y,CHR$(CH);
3110 ' FILL IN LINE
3120 IF YL=YH THEN 3160
3130 FOR K=YL+1 TO YH-1
3140 PRINT@480+X-32*K,CHR$(138);
3150 NEXT K
3160 NEXT J
3170 '
3180 ' CLEAR PLOT VARIABLES
3190 FOR J=1TO10:Y(J)=0:NEXT J
3200 '
3210 ' START OF MAIN PLOT LOOP
3220 FOR FD=1 TO FB
3230 FOR J=1 TO 10
3240 IF PEEK(STATUS) THEN 3260
3250 IF INKEY$="E" THEN RETURN ELSE 3240
3260 X=3*J:DA=B1+B2*FD+B3*VAL(DW$(J)):REM POSITION AND  DATA
ADDRESS
3270 Y=256*PEEK(DA)+PEEK(DA+1):REM READ DATA
3280 Y=INT(100*Y/B):REM CONVERT TO %
3290 PRINT@X+62,INT(Y);:REM PUT UP PERCENTAGE
```

```
3300 GOSUB 2610
3310 IF INKEY$<>"" THEN RETURN
3320 NEXT J,FD
3330 GOTO 3220
3340 '
3350 '
3360 REM ************************************
3370 REM *                                  *
3380 REM *    PLOT DATA WORD ROUTINE        *
3390 REM *                                  *
3400 REM * THIS ROUTINE PLOTS 29 SEQ-       *
3410 REM * ENTIAL POINTS OF ONE DATA        *
3420 REM * WORD IN DW$(LM).  LM IS THE      *
3430 REM * POINTER FROM THE ENG DATA        *
3440 REM * PAGE.                            *
3450 REM *                                  *
3460 REM ************************************
3470 '
3480 ' FIRST DRAW CHART
3490 CLS:PRINT" PLOT FOR DATA WORD# ";DW$(LM);" ";N$(LM);
3500 PRINT STRING$(32,"*");
3510 PRINT "%"
3520 FOR J=90 TO 0 STEP -10
3530 PRINT RIGHT$(STR$(J),2);CHR$(154):REM VERT LINE
3540 NEXT J
3550 PRINT@416,"  ";CHR$(156);:REM LOWER LEFT CORNER
3560 FOR J=1TO29
3570 PRINT CHR$(156);
3580 NEXT J
3590 '
3600 A$="c)ont, T)RACE":REM DEFAULT
3610                PRINT@448,"RATE=";:PRINT              USING
"###.##";(256*PEEK(CALRATE)+PEEK(CALRATE+1))/16;:
     PRINT"MS      ";A$
3620 PRINT"E)XIT, _^ TO SCROLL, PG    OF34";
3630 IF LEFT$(A$,1)="C" THEN 3860:REM MUST BE "t" FOR TRACE
3640 '
3650 ' MAIN PLOT LOOP FOR cont
3660 DA=B1+B2+B3*VAL(DW$(LM)):REM FIRST ADDRESS
3670 DB=DA:REM KEEP DB=START ADDRESS
3680 DC=B1+B2*FB+B3*VAL(DW$(LM)):REM LM  HAS  SELECTED  LINE
NUMBER
3690 FOR J=1TO29:REM X POSITION FOR POINT
3700 IF PEEK(STATUS) THEN 3720
3710 IF INKEY$="E" THEN RETURN ELSE 3700
3720 X=J+2
3730 Y=256*PEEK(DA)+PEEK(DA+1):REM GET DATA
3740 Y=INT(100*Y/B):REM CONVERT TO %
3750 GOSUB 2610:REM PLOT THE POINT
3760 KY$=INKEY$:IF KY$<>"" THEN 3810
3770 DA=DA+B2:REM SKIP TO NEXT FRAME
3780 IF DA>DC THEN DA=DB:REM START OVER FROM FIRST FRAME
```

```
3790 NEXT J
3800 GOTO 3690
3810 IF KY$="E" THEN RETURN
3820 IF KY$="T" THEN A$="C)ONT, t)race"
3830 GOTO 3610
3840 '
3850 ' MAIN PLOT ROUTINE FOR trace
3860 J=1:REM USED FOR X POSITIONING
3870 K=0:REM USED FOR PAGE COUNTING
3880 A=0:REM USED FOR ADDRESS COUNTING
3885 POKE WC,192:POKE WC+1,0:REM SET ADDRESS TO BEGINNING OF
$C000
3890 X=3:REM START AT LEFT SIDE
3900  POKE VLID,255:POKE VLID+1,VAL(DW$(LM)):POKE VLID,0:REM
START TRACE
3910 PRINT@503,1;
3920 DA=BUFFER+A+K:REM DATA ADDRESS
3930 IF TEST=1 THEN 3960:REM USED TO TEST TRACE  MODE.  TEST
IS          SET IN LINE 10
3940  IF  PEEK(WC)*256+PEEK(WC+1)+8192  >=DA  THEN  3960:REM
PROCESSOR IS AHEAD OF DISPLAY
3950 KY$=INKEY$: IF KY$="" THEN 3940 ELSE  4030:REM  PROCESS
KEY
3960 Y=256*PEEK(DA)+PEEK(DA+1)
3970 Y=INT(100*Y/B):REM CONVERT TO %
3980 GOSUB 2610:REM PLOT POINT
3990 J=J+1:A=A+2:X=X+1
4000 IF A<>58 THEN 3920:REM PLOT NEXT POINT
4010 A=0:X=3:J=1:REM REWIND TO LEFT SIDE
4020 KY$=INKEY$: IF KY$="" THEN 4020
4030 IF KY$="E" THEN POKE VLID,255: FOR I=1TO100:
     NEXT  I:POKE  VLID+1,0:POKE  VLID,0:RETURN:REM STOP AND
START        OVER
4040 IF KY$="C" THEN POKE  VLID,255:FOR  I=1TO100:NEXTI:POKE
VLID+1,0:POKE VLID,0:GOTO 3600
4045 IF KY$="T" THEN 3860:REM RESTART TRACE FROM BEGINNING
4050 IF KY$<>CHR$(8) THEN 4080:REM LT ARROW
4060 K=K-58:  IF K<0 THEN K=1914
4070 GOTO 4090:REM DO NEXT PAGE
4080  K=K+58:IF  K>1914  THEN K=0:REM ANY OTHER KEY ADVANCES
PAGE
4090 PRINT@503,K/58+1;:REM PRINT PAGE NUMBER
4100 GOTO 3920
4110 REM
4120 REM  ************************
4130 REM  *                      *
4140 REM  *     INPUT ROUTINE    *
4150 REM: * SP= SCREEN POSITION *
4160 REM: * RETURNS STRING IN A$*
4170 REM: * N = MAX CHARACTERS  *
4180 REM: * I = TEMP COUNTER    *
4190 REM: * PSN = LOCAL POSITION*
```

```
4200 REM  * LN = LINE OF ENTRY  *
4210 REM: *                     *
4220 REM: ***********************
4230 REM
4240 SAME=0:PSN=1:PRINT@SP(LN),"["+A$;:PRINT@SP(LN)+N+1,"]";
4250 REM MAKE FLASHING CURSOR
4260 IF PSN=N+1 THEN 4310:REM END OF LINE
4270 X=SP(LN)+PSN+1024:REM DIRECT READ OF SCREEN
4280 CH=PEEK(X)
4290 POKE X,109:FOR I=1TO TIME:NEXT  I:POKE  X,CH:FOR  I=1TO
TIME:NEXT I
4300 REM 109 = "-"
4310 A$=INKEY$: IF A$="" THEN 4260
4320 KY=ASC(A$)
4330 IF KY=12 OR KY=13 THEN 4410:REM CLEAR OR ENTER
4340 IF KY=8 AND PSN =1 THEN 4310:REM LEFT ARROW
4350 IF KY=8 THEN PSN=PSN-1:GOTO4310:REM LEFT ARROW
4360 IF KY=10 OR KY=94 THEN 4420:REM DOWN OR UP
4370  IF  PSN  <1 OR PSN >N THEN 4310 ELSE PRINT@SP(LN)+PSN,
A$;
4380 IF KY=9 THEN 4390 ELSE SAME=1:REM NEW DATA ENTERED
4390 PSN=PSN+1
4400 GOTO4310
4410 REM COLLECT CHARACTERS INTO A$
4420 A$=""
4430 FOR PSN=1  TO  N:X=PEEK(SP(LN)+PSN+1024):IF  X>95  THEN
X=X-64
4440 A$=A$+CHR$(X):NEXT PSN
4450   PRINT@SP(LN),"  ";:PRINT@SP(LN)+N+1,"  ";:REM  REMOVE
BRACKETS
4460 RETURN
4470 REM
4480 REM
4490 REM ************************
4500 REM *                      *
4510 REM *  DISK INPUT ROUTINE  *
4520 REM *                      *
4530 REM *  RECORD # IN PN      *
4540 REM *  RECORD NAME IN NAME$*
4550 REM *  RETURNS DATA IN A$  *
4560 REM *                      *
4570 REM ************************
4580 IF REC=0 THEN RETURN
4590 OPEN"D", #1, NAME$,SIZE
4600 GET#1, REC
4610 INPUT #1,A$
4620 CLOSE #1
4630 RETURN
4640 REM ************************
4650 REM *                      *
4660 REM *  DISK OUTPUT ROUTINE *
4670 REM *                      *
```

```
4680 REM *  RECORD # IN PN       *
4690 REM *  RECORD NAME IN NAME$*
4700 REM *  DATA IN A$           *
4710 REM *                       *
4720 REM *************************
4730 IF REC=0 THEN RETURN
4740 OPEN "D", #1, NAME$, SIZE
4750 WRITE #1,A$
4760 PUT #1,REC
4770 CLOSE #1
4780 RETURN
4790 REM
4800 REM ************************
4810 REM *                      *
4820 REM * READ IN RECORD AND  *
4830 REM * CONVERT TO VARIABLES*
4840 REM *                      *
4850 REM ************************
4860 REM
4870 REM
4880 REM
4890 REC=2*PN-1:REM GET FIRST HALF
4900 GOSUB 4580:REM THEN READ IT IN
4910 B=2^VAL(B$)-1:REM ABSOLUTE MAX VALUE FOR WORD
4920 FOR I=1TO10
4930 IF I<6 THEN X=(I-1)*43+1 ELSE X=(I-6)*43+1
4940 IF I=6 THEN REC=2*PN:GOSUB 4580:REM GET  NEXT  HALF  OF
RECORD
4950 DW$(I)=MID$(A$,X,3):REM DATA WORD NUMBER
4960 N$(I)=MID$(A$,X+3,7):REM NAME
4970 P$(I)=MID$(A$,X+10,3):REM PARAMETER NUMBER
4980 U$(I)=MID$(A$,X+13,5):REM UNITS
4990 MN(I)=VAL(MID$(A$,X+18,6)):REM RANGE MINIMUM
5000 MX(I)=VAL(MID$(A$,X+24,6)):REM RANGE MAXIMUM
5010 AL(I)=VAL(MID$(A$,X+30,6)):REM ALARM LOW
5020 AH(I)=VAL(MID$(A$,X+36,6)):REM ALARM HIGH
5030 D$(I)=MID$(A$,X+42,1):REM DISPLAY TYPE
5040 '
5050 '
5060  '  CALCULATE SCALE FACTOR MAPPING 0-MAX VALUE FOR WORD
TO
5070 ' RANGE MIN AND MAX
5080 SCALE(I)=(MX(I)-MN(I))/B:  REM  SCALE  FOR  ENGINEERING
UNITS
5090 NEXT I
5100 RETURN
5110 REM ************************
5120 REM *                      *
5130 REM *     STATUS OUTPUT    *
5140 REM *        ROUTINE       *
5150 REM *                      *
5160 REM ************************
```

```
5170 A$=" STATUS PAGE. KEY TO EXIT":TP=1:GOSUB 2250
5180 PRINT"                  USER  CALCULATED"
5190 PRINT"BITS/WORD          ";B$;"       ----"
5200 PRINT"WORDS/FRAME        ";W$;"       ----"
5210 PRINT"PCM TYPE           ";TY$
5220 IF RA$="auto" THEN A$="AUTO" ELSE A$=RA$:REM MAKE CAPS
5230 PRINT"BIT RATE           ";A$
5240 PRINT"PCM ORDER          ";O$;"       ----"
5250 PRINT"PCM POLARITY       ";PO$;"   ----"
5260 PRINT"SYNC ERRORS        ----"
5270 PRINT"PCM HEALTH         ----"
5280 PRINT"SYNC WORDS 1 - ";SW$(1)
5290 PRINT" ";CHR$(94);" INVERSE 2 - ";SW$(2)
5300 PRINT"IF IN SYNC 3 - ";SW$(3)
5310 PRINT@217,CHR$(PEEK(RPCMTYPE+1));
5315  X= 256*PEEK(CALRATE)+PEEK(CALRATE+1):IF X<16.0001 THEN
X=16.0001:REM MAX RATE FOR DISPLAY
5320 PRINT@245,"";:PRINT USING "###.##";16000/X;:PRINT"KHZ";
5330 PRINT@344,256*PEEK(EROR)+PEEK(EROR+1)
5340 PRINT@376,PEEK(CORRECTION+1)
5350 IF PEEK(STATUS)=0 THEN A$="SYNC" ELSE A$="sync"
5360 PRINT@384,A$;
5370 A$=INKEY$:IF A$="" THEN 5310
5380 RETURN
5390 GOTO 5390
5400 REM ************************
5410 REM *                      *
5420 REM *       MAIN MENU      *
5430 REM *        ROUTINE       *
5440 REM *                      *
5450 REM ************************
5460 REM
5470 A$="PCM DISPLAY MODULE":TP=1:GOSUB2250:REM OUTPUT  MENU
PAGE
5480 PRINT@128,"S)TATUS PAGE"
5490 PRINT"B)EGIN DISPLAY PROCESSING"
5500 PRINT"Q)UIT"
5510 PRINT:INPUT "WHICH ONE->";A$
5520 IF A$="Q" THEN RUN"MAIN"
5530 IF A$="B" THEN RETURN
5540 IF A$="S" THEN GOSUB 5170
5550 GOTO 5470
5560 REM ***********************
5570 REM *                     *
5580 REM *  START OF PROGRAM   *
5590 REM *                     *
5600 REM ***********************
5610 REM
5620 GOSUB 920: REM INIT
5625 GOSUB 1680: REM LOAD AND START PROCESSOR
5630 GOSUB 5470:REM MENU PAGE
5640 GOSUB 2430: REM OUTPUT ENG DATA PAGE
```

```
5650 A$="    "
5660 LN=0:N=2:GOSUB 4240:REM LINE INPUT ROUTINE
5670 IF VAL(A$)=0 OR VAL(A$)>VAL(MR$) THEN 5650
5680 PN=VAL(A$):REM PN=ABSOLUTE RECORD #
5690  GOSUB  4890:REM  READ  IT  IN  AND  CONVERT TO DISPLAY
VARIABLES
5700 FOR I=1TO10
5710 X$=U$(I)
5720 IF D$(I)="H" THEN X$="HEX   "
5730 IF D$(I)="O" THEN X$="OCT   "
5740 IF D$(I)="B" THEN X$="BIN   "
5750 IF D$(I)="S" THEN X$="SWI   "
5760   PRINT@SP(I),DW$(I);"    ";N$(I);"                    ";X$;"
";D$(I);
5770 NEXT I
5780 '
5790 '
5800 '
5810 ' START OF MAIN LOOP
5820 '
5830 ' PUT AS MUCH IN VARIABLE FORM TO INCREASE SPEED
5840 I=1:FC=1:REM FIRST DATA WORD, FIRST FRAME
5850 B1=BUFFER-2*VAL(W$)-2
5860 B2=2*VAL(W$)
5870 B3=2
5880 POKE YP(LM),PEEK(YP(LM))-64:REM INVERSE TYPE IN LINE LM
5890 '
5900 ' MAIN LOOP
5910 '
5920 IF PEEK(STATUS) THEN 5940
5930  IF INKEY$="E" THEN 5630 ELSE 5920:REM WAIT FOR SYNC OR
EXIT      KEY
5940 DA=B1+B2*FC+B3*VAL(DW$(I))
5950 PRINT@VP(I)-1,"        ";:REM CLEAR LINE BEFORE PRINTING
5960 IF D$(I)="H" THEN PRINT@VP(I),"";: X=INT(DA/256):  POKE
VO,DA-256*X:POKE VH,X:A=USR1(1):GOTO 6060
5970  IF  D$(I)="O" THEN PRINT@VP(I),"";:X=INT(DA/256): POKE
VO,DA-256*X:POKE VH,X:A=USR1(2):GOTO 6060
5980 IF D$(I)="B" THEN  PRINT@VP(I),"";:X=INT(DA/256):  POKE
VO,DA-256*X:POKE          VH,X:POKE          VO+2,VAL(B$):POKE
VH+2,0:A=USR1(3):GOTO 6060
5990 IF D$(I)<>"S" THEN 6020
6000 IF PEEK(DA)=0 AND PEEK(DA+1)=0 THEN  PRINT@VP(I),"OFF";
ELSE      PRINT @VP(I),"ON";
6010 GOTO 6060
6020 X=256*PEEK(DA)+PEEK(DA+1):REM READ DATA
6030 X=SCALE(I)*X+MN(I)
6040 PRINT@VP(I)-1,"";
6050 PRINT USING "#####.##";X;
6060 A$=INKEY$:IF A$<>"" THEN GOTO 6130
6070 ' SKIP FRAME IF SAME LINE TO SEE TWO SEQUENTIAL VALUES
6080  IF  N$(I)=N$(I+1)  AND  D$(I)=D$(I+1) THEN FC=FC+1:REM
```

```
     FRAME
6090 I=I+1:IF I=11 THEN I=1:FC=FC+1:IF FC>=FB+1 THEN FC=1
6100 GOTO 5920
6110 '
6120 ' KEY DECODE ROUTINE
6130 IF ASC(A$)=9 THEN PN=PN+1: IF PN>VAL(MR$) THEN PN=1:REM
RT        ARROW
6140 IF ASC(A$)=8 THEN PN=PN-1: IF PN<1 THEN PN=VAL(MR$):REM
LEFT ARROW
6150 PRINT@27,PN:REM DISPLAY IT
6160 IF VAL(A$)<>0 THEN 5660:REM A  NUMBER
6170 IF ASC(A$)=8 OR ASC(A$)=9 THEN 5690:REM ALREADY DECODED
SO        GO BACK
6180 IF ASC(A$)<>94 THEN 6220:REM UP ARROW
6190 POKE YP(LM),PEEK(YP(LM))+64:REM NORMAL CHAR
6200 LM=LM-1:IF LM<1 THEN LM=10:REM  DECREMENT  SELECT  LINE
POINTER
6210  POKE  YP(LM),PEEK(YP(LM))-64:REM  MAKE  SELECTED  TYPE
INVERSE
6220 IF ASC(A$)<>10 THEN 6260:REM DOWN ARROW
6230 POKE YP(LM),PEEK(YP(LM))+64:REM MAKE OLD SELECTED  TYPE
NORMAL
6240 LM=LM+1:IF LM>10 THEN LM=1:REM INCREMENT LINE POINTER
6250 POKE YP(LM),PEEK(YP(LM))-64:REM MAKE NEW TYPE INVERSE
6260 IF A$="E" THEN GOTO 5630:REM EXIT
6270 IF A$<>"T" THEN 6370:REM TOGGLE TYPE
6280 '
6290 ' DO THE TYPE TOGGLE FUNCTION
6300          IF          PEEK(YP(LM))=4          THEN          POKE
YP(LM),8:D$(LM)="H":PRINT@VP(LM)+5,"          HEX       ";:GOTO
6390:REM DEC TO HEX
6310          IF          PEEK(YP(LM))=8          THEN          POKE
YP(LM),15:D$(LM)="O":PRINT@VP(LM)+5,"         OCT       ";:GOTO
6390:REM HEX TO OCTAL
6320          IF          PEEK(YP(LM))=15          THEN          POKE
YP(LM),2:D$(LM)="B":PRINT@VP(LM)+6,"         BIN       ";:GOTO
6390:REM OCTAL TO BINARY
6330          IF          PEEK(YP(LM))=2          THEN          POKE
YP(LM),19:D$(LM)="S":PRINT@VP(LM)+5,"         SWI       ";:GOTO
6390:REM BINARY TO SWITCH
6340          IF          PEEK(YP(LM))=19          THEN          POKE
YP(LM),4:D$(LM)="D":PRINT@VP(LM)+5,"     ";U$(LM);"    ";:REM
SWITCH TO DECIMAL
6350 '
6360 ' CALL BAR CHART ROUTINE IF REQ
6370  IF A$="B" THEN GOSUB 2750:GOSUB 2430:PRINT@27,PN;:GOTO
5700:REM DO BAR FUNCTION THEN ENG PAGE
6380 IF A$="P" THEN GOSUB 3490:GOSUB  2430:PRINT@27,PN;:GOTO
5700:REM  DO  PLOT  FUNCTION  THEN  RETURN  TO ENG PAGE UPON
KEYPRESS
6390 GOTO 6090
```

Appendix D

REAL-TIME FLIGHT TEST PCM DATA ACQUISITION MONITOR

APPLICATION NOTES

BY

JOHN R. CROASDALE, LTC, USAF

MOTOROLA

SEMICONDUCTORS

3501 ED BLUESTEIN BLVD . AUSTIN. TEXAS 78721

Advance Information

MC68000L4
(4 MHz)
MC68000L6
(6 MHz)
MC68000L8
(8 MHz)
MC68000L10
(10 MHz)
MC68000L12
(12.5 MHz)

HMOS
(HIGH-DENSITY, N-CHANNEL, SILICON-GATE DEPLETION LOAD)

16-BIT MICROPROCESSOR

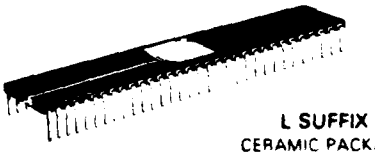
16-BIT MICROPROCESSING UNIT

Advances in semiconductor technology have provided the capability to place on a single silicon chip a microprocessor at least an order of magnitude higher in performance and circuit complexity than has been previously available. The MC68000 is the first of a family of such VLSI microprocessors from Motorola. It combines state-of-the-art technology and advanced circuit design techniques with computer sciences to achieve an architecturally advanced 16-bit microprocessor.

The resources available to the MC68000 user consist of the following:

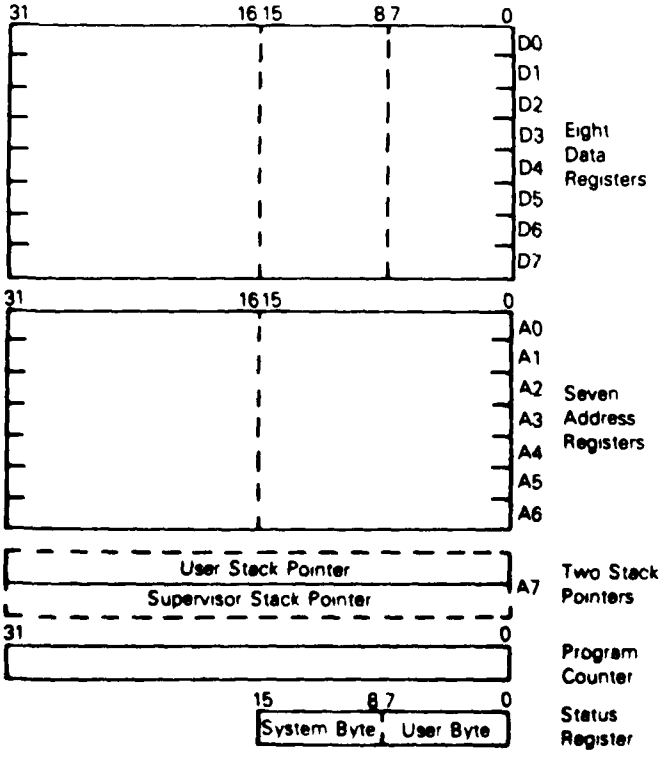
- 32-Bit Data and Address Registers
- 16 Megabyte Direct Addressing Range
- 56 Powerful Instruction Types
- Operations on Five Main Data Types
- Memory Mapped I/O
- 14 Addressing Modes

As shown in the programming model, the MC68000 offers seventeen 32-bit registers in addition to the 32-bit program counter and a 16-bit status register. The first eight registers (D0-D7) are used as data registers for byte (8-bit), word (16-bit), and long word (32-bit) data operations. The second set of seven registers (A0-A6) and the system stack pointer may be used as software stack pointers and base address registers. In addition, these registers may be used for word and long word address operations. All seventeen registers may be used as index registers.

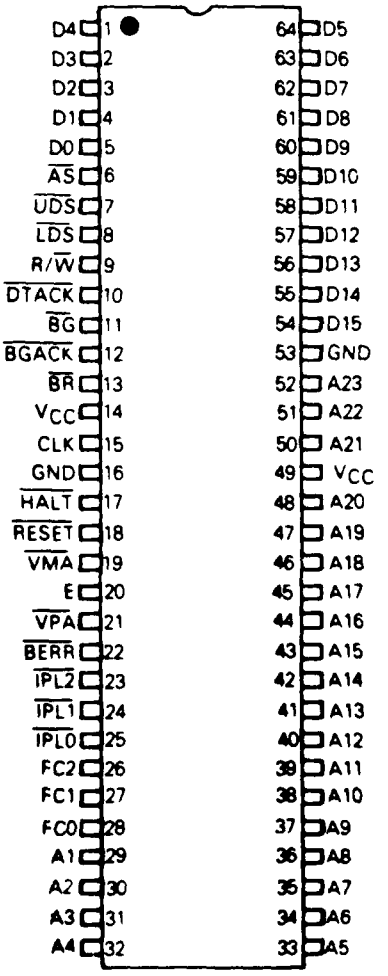


L SUFFIX
CERAMIC PACKAGE
CASE 746

PROGRAMMING MODEL



PIN ASSIGNMENT



INSTRUCTION SET OVERVIEW

The MC68000 instruction set is shown in Table 10. Some additional instructions are variations, or subsets, of these and they appear in Table 11. Special emphasis has been given to the instruction set's support of structured high-level languages to facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic and expanded operations (through traps).

TABLE 10 – INSTRUCTION SET

Mnemonic	Description
ABCD	Add Decimal with Extend
ADD	Add
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
B_{CC}	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Compare
DB_{CC}	Test Condition, Decrement and Branch
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Negate
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PEA	Push Effective Address
RESET	Reset External Devices
ROL	Rotate Left without Extend
ROR	Rotate Right without Extend
ROXL	Rotate Left with Extend
ROXR	Rotate Right with Extend
RTE	Return from Exception
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
S_{CC}	Set Conditional
STOP	Stop
SUB	Subtract
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink

TABLE 11 – VARIATIONS OF INSTRUCTION TYPES

Instruction Type	Variation	Description
ADD	**ADD**	Add
	ADDA	Add Address
	ADDQ	Add Quick
	ADDI	Add Immediate
	ADDX	Add with Extend
AND	**AND**	Logical And
	ANDI	And Immediate
CMP	**CMP**	Compare
	CMPA	Compare Address
	CMPM	Compare Memory
	CMPI	Compare Immediate
EOR	**EOR**	Exclusive Or
	EORI	Exclusive Or Immediate
MOVE	**MOVE**	Move
	MOVEA	Move Address
	MOVEQ	Move Quick
	MOVE from SR	Move from Status Register
	MOVE to SR	Move to Status Register
	MOVE to CCR	Move to Condition Codes
	MOVE USP	Move User Stack Pointer
NEG	**NEG**	Negate
	NEGX	Negate with Extend
OR	**OR**	Logical Or
	ORI	Or Immediate
SUB	**SUB**	Subtract
	SUBA	Subtract Address
	SUBI	Subtract Immediate
	SUBQ	Subtract Quick
	SUBX	Subtract with Extend

CMOS DUAL PORT RAM 8K (1K x 8 BIT)

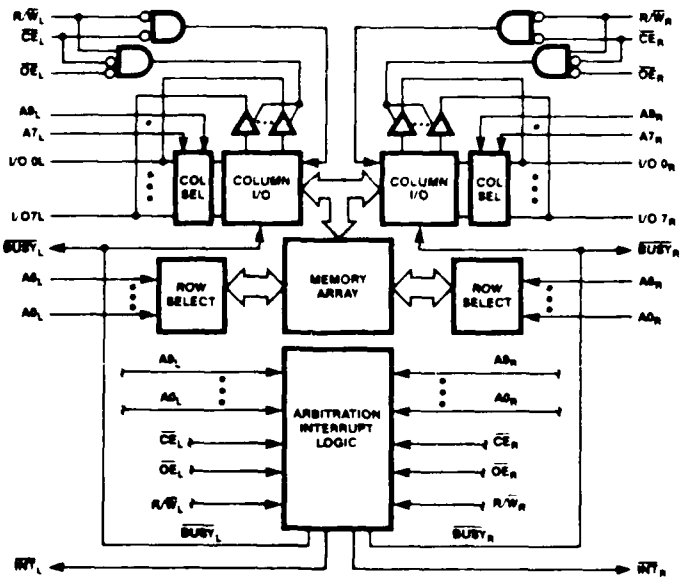
ADVANCE INFORMATION
IDT7130S
IDT7130L

MILITARY / INDUSTRIAL / COMMERCIAL TEMPERATURE RANGES

FEATURES:

- High-speed access
  - Military/Industrial—100/120ns (max.)
  - Commercial—90/100ns (max.)
- Low-power operation
  - IDT7130S
    - Active—325mW (typ.)
    - Standby—200μW (typ.)
  - IDT7130L
    - Active—325mW (typ.)
    - Standby—50μW (typ.)
- CEMOS™ II process virtually eliminates alpha particle induced soft errors
- On-chip port arbitration logic
- $\overline{INT}$ and $\overline{BUSY}$ flags
- Fully asynchronous operation from either port
- Battery backup operation—2V data retention
- Single 5V ± 10% power supply
- TTL compatibility
- Fully static operation
- Three state output
- Military product 100% screened to MIL-STD-883, Class B

FUNCTIONAL BLOCK DIAGRAM



DESCRIPTION:

The IDT7130 is a CMOS 1Kx8 high-speed Dual Port Static RAM. It is fabricated using IDT's high performance CEMOS™ II technology. This state-of-the-art technology, combined with innovative circuit design techniques, provides low power alternatives to fast NMOS-type memories.

The IDT7130 provides two ports with separate controls, address and I/O that permit independent access for reads or writes to any location in memory. The IDT7130 has an automatic power-down feature controlled by $\overline{CE}$. The $\overline{CE}$ controls on-chip power-down circuitry that permits the respective port to go into a standby mode when not selected ($\overline{CE}$ high).

The interrupt flag ($\overline{INT}$) permits communication between ports or systems. If the user chooses to use the interrupt function, a memory location (mail box or message center) is assigned to each port. The left port interrupt flag ($\overline{INT}_L$) is set when the right port writes to memory location 3FE. The left port clears the interrupt by reading address location 3FE. Likewise, the right port interrupt flag ($\overline{INT}_R$) is set when the left port writes to memory location 3FF and to clear the interrupt flag ($\overline{INT}_R$), the right port must read the memory location 3FF. The message (8-bits) at 3FE or 3FF is user-defined. If the interrupt function is not used, address locations 3FE and 3FF are not used as mail boxes but as part of the random access memory.

The $\overline{BUSY}$ flags are provided for the situation when both ports simultaneously access the same memory location. When this situation occurs, on-chip arbitration logic will determine which port has access and sets the $\overline{BUSY}$ flag of the delayed port. $\overline{BUSY}$ is set at speeds that permit the processor to hold the operation and its respective address and data. The delayed port will have access when $\overline{BUSY}$ goes inactive.

Access times as fast as 90ns are available with typical operating power of only 325mW. The low power (L) version also offers a battery backup data retention capability where the circuit typically consumes only 2.5μW off a 2V battery.

The IDT7130 Dual Port RAM is designed to take into consideration the 2Kx8 Dual Port RAM. The $\overline{INT}$ function is not available on the 2Kx8 organization because the pin is needed for address A_{10}. Other than this difference, the 2Kx8 is pin-for-pin compatible with the 1Kx8. The 2Kx8 Dual Port RAM, IDT7132, doubles the density for the same package size, thus improving the system over all reliability and power requirements. System reliability will be improved by halving the part count and the system power requirement and, as an added side benefit, valuable board space and power is freed up for other system requirements.

The IDT7130 is packaged in either a 48-pin DIP or a 48-pin leadless chip carrier. Military parts are 100% processed in compliance to the test methods of MIL-STD-883, Method 5004, making them ideally suited to military temperature applications demanding the highest level of performance and reliability.

TOSHIBA MOS MEMORY PRODUCTS

2048 WORD X 8 BIT STATIC RAM

N CHANNEL SILICON GATE DEPLETION LOAD

TMM2016P
TMM2016P-1

DESCRIPTION

The TMM2016P is a 16384 bit static random access memory organized as 2048 words by 8 bits and operates from a single 5V power supply. Common 8-bit input/output, output enable ($\overline{OE}$) and pin-compatibility with 2716 type EPROM (TMM323C) allow a wide application in microprocessor peripheral memory.

In memory expansion, low power application is possible by using the chip select input ($\overline{CS}$). When $\overline{CS}$ is in V_{IH} level, the device is in low power standby mode, and the operating current is reduced to 7mA from 60mA (Typical).

TMM 2016P is fabricated with ion implanted N-channel silicon gate technology. This technology provides high performance and high reliability. The chip is moulded in a 24 pin standard plastic package, 0.6 inch in width.

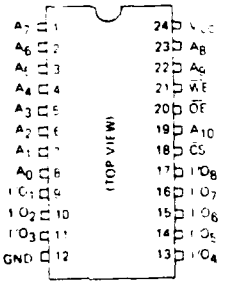
FEATURES

- Pin compatible with 2716 type EPROM
- Single 5V supply – V_{CC} = 5V ± 10%
- Access time and current

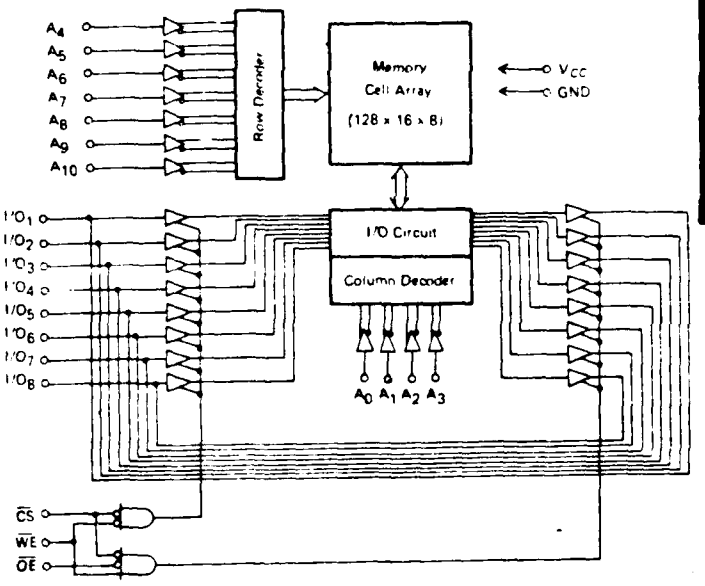
	TMM2016P	TMM2016P-1
Access time (MAX.)	150 ns	100 ns
Operating current (MAX.)	100mA	120mA
Standby current (MAX.)	15mA	15mA

- Power down feature – $\overline{CS}$
- Output buffer control – $\overline{OE}$
- Easy memory expansion – $\overline{CS}$
- Static operation – No clock or timing strobe required
- Directly TTL compatible – All inputs and outputs
- Common data input and output
- Three state outputs – Wired OR capability
- Inputs protected – All inputs have protection against static charge.

PIN CONNECTION



BLOCK DIAGRAM



PIN NAMES

SYMBOL	NAME
$A_0 \sim A_3$	Column Address Inputs
$A_4 \sim A_{10}$	Row Address Inputs
$\overline{CS}$	Chip Select Input
$\overline{WE}$	Write Enable Input
$I/O_1 \sim I/O_8$	Data Input/Output
$\overline{OE}$	Output Enable Input
V_{CC}	Power (5V)
GND	Ground

Toshiba

MEMORY

TMM2016P
TMM2016P-I

MAXIMUM RATINGS

SYMBOL	ITEM	RATING	UNIT
V_{CC}	Power Supply Voltage	−0.5 ~ 7.0	V
V_{IN}, V_{OUT}	Input and Output Voltage	−0.5 ~ 7.0	V
T_{OPR}	Operating Temperature	0 ~ 70	°C
T_{STG}	Storage Temperature	−55 ~ 150	°C
T_{SOLDER}	Soldering Temperature · Time	260 · 10	°C · sec
P_D	Power Dissipation (Ta = 70°C)	1.0	W

D.C. RECOMMENDED OPERATING CONDITIONS (Ta = 0 ~ 70°C)

SYMBOL	PARAMETER	MIN	TYP	MAX	UNIT
V_{IH}	Input High Voltage	2.2	—	V_{CC} + 1.0	V
V_{IL}	Input Low Voltage	−0.5	—	0.8	V
V_{CC}	Supply Voltage	4.5	5.0	5.5	V

D.C. CHARACTERISTICS (Ta = 0 ~ 70°C, V_{CC} = 5V ± 10%)

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNIT
I_{IL}	Input Leakage Current	V_{IN} = 0 ~ 5.5V		−	−	±10	μA
I_{OH}	Output High Current	V_{OUT} = 2.4V		−1.0	−	−	mA
I_{OL}	Output Low Current	V_{OUT} = 0.4V		2.1	−	−	mA
V_{OH}	Output High Voltage	I_{OUT} = −1.0mA		2.4	−	−	V
V_{OL}	Output Low Voltage	I_{OUT} = 2.1mA		−	−	0.4	V
I_{LO}	Output Leakage Current	$\overline{CS}$ = V_{IH} or $\overline{WE}$ = V_{IL} or $\overline{OE}$ = V_{IH} V_{OUT} = 0 ~ V_{CC}		−	−	±10	μA
* I_{SBP}	Peak Power-on Current	$\overline{CS}$ = V_{CC}, I_{OUT} = 0mA during power on		−	−	30	mA
I_{SB}	Standby Current	$\overline{CS}$ = V_{IH}, I_{OUT} = 0mA		−	7	15	mA
I_{CC}	Operating Current	$\overline{CS}$ = V_{IL}, I_{OUT} = 0mA	TMM2016P	−	−	100	mA
			TMM2016P-1	−	−	120	

* Note I_{CC} exceeds I_{SB} maximum during power on. A pull up resistor to V_{CC} on the $\overline{CS}$ input is required to keep the device deselected, otherwise power-on current approaches I_{CC} active.

* CAPACITANCE (Ta = 25°C, f = 1 MHz)

SYMBOL	PARAMETER	CONDITIONS	MAX	UNIT
C_{IN}	Input Capacitance	V_{IN} = A.C. Ground	5	pF
C_{OUT}	Output Capacitance	V_{OUT} = A.C. Ground	10	pF

* Note This parameter is periodically sampled and is not 100% tested.

TMM2016P
TMM2016P-1

A.C. CHARACTERISTICS (Ta = 0 ~ 70°C, V_{CC} = 5V ± 10%)

READ CYCLE

SYMBOL	PARAMETER	TMM2016P		TMM2016P-1		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{RC}	Read Cycle Time	150	–	100	–	ns
t_{ACC}	Address Access Time	–	150	–	100	ns
t_{CO}	Chip Select Access Time	–	150	–	100	ns
t_{OE}	Output Enable Time	–	55	–	35	ns
t_{OH}	Output Hold Time from Address Change	10	–	10	–	ns
t_{CLZ}	Output in Low Z from $\overline{CS}$	10	–	10	–	ns
t_{CHZ}	Output in High Z from $\overline{CS}$	–	55	–	40	ns
t_{OLZ}	Output in Low Z from $\overline{OE}$	5	–	5	–	ns
t_{OHZ}	Output in High Z from $\overline{OE}$	–	50	–	35	ns
t_{PU}	Chip Selection to Power up Time	0	–	0	–	ns
t_{PD}	Chip Deselection to Power down Time	–	60	–	50	ns

WRITE CYCLE

SYMBOL	PARAMETER	TMM2016P		TMM2016P-1		UNIT
		MIN.	MAX.	MIN.	MAX.	
t_{WC}	Write Cycle Time	150	–	100	–	ns
t_{CW}	Chip Selection to End of Write	120	–	90	–	ns
t_{AS}	Address Set up Time	20	–	20	–	ns
t_{WP}	Write Pulse Width	100	–	70	–	ns
t_{WR}	Write Recovery Time	10	–	10	–	ns
t_{DS}	Data Set up Time	60	–	40	–	ns
t_{DH}	Data Hold Time	15	–	10	–	ns
t_{WLZ}	Output in Low-Z from $\overline{WE}$	5	–	5	–	ns
t_{WHZ}	Output in High-Z from $\overline{WE}$	–	50	–	35	ns

A.C. TEST CONDITIONS

Input Pulse Levels	0 ~ 3.5 V
Input Rise and Fall Times	10ns
Input and Output Timing Reference Levels	1.5V
Output Load	See Note

Note Output Load – 1TTL Gate and C_L = 100pF
(Including scope and jig)

Appendix E

REAL-TIME FLIGHT TEST PCM DATA ACQUISITION MONITOR

PARTS LIST

BY

JOHN R. CROASDALE, LTC, USAF

Parts List

PCM Processor

Integrated Circuits

Quantity	IC number	Remarks
1	MC-68000	Motorola 8 Mhz Microprocessor
2	IDT 7130S	Dual Port RAM 120ns
2	TMM 2016P	2K by 8 RAM, 150ns
2	TMM 2716	EPROM 450ns
1	NCT070C	Saronix 32Mhz clock hybrid
2	74LS00	Quad 2 input NAND gate
2	74LS02	Quad 2 input NOR gate
2	74LS04	Hex inverter
1	74HC04	CMOS hex inverter
2	74LS08	Quad 2 input AND gate
1	74LS14	Hex Schmitt trigger
1	74LS21	Dual 4 input AND gate
2	74LS30	8 input NAND gate
1	74HC32	Quad 2 input OR gate
1	74LS32	
2	74LS74	Dual D type flip flop
1	74HC75	4 bit bistable latchs
1	74LS93	Binary counter
1	74121	Monostable multivibrator with clear
1	74123	Dual monostable multivibrator
1	74LS138	3 to 8 line decoder
2	74LS175	Quad D type flip flop
9	74LS193	Up/Down binary counter
1	74LS279	Quad SR latches
2	74LS299	8 bit bidirectional shift register
4	74HC374	Octal D type flip flops
2	74LS374	
2	74LS393	Dual 4 bit binary counters
1	555	Timer

Other Parts

1	Single pole signle throw switch 2 amps, 5 volts minimum
1	Double pole double throw switch 2 ampa, 5 volts minimum
20	.1 microfarad dipped ceramic capacitors
1	10 microfarad tantalum capacitor
1	.033 microfarad mica capacitor
1	27K ohm 1/4 watt resistor
1	18K ohm 1/4 watt resistor
1	470 ohm 1/4 watt resistor
2	general purpose signal diodes
1	Wire Wrap board

Appendix F

Acronym Listing

The listing below shows some of the common acronyms used in this thesis to be used for reference for the reader as needed.

ARIA ---------- Advanced Range Instrumented Aircraft
AS ---------- Address Strobe
BAS ---------- Short for BASIC appended to BASIC files
BIN ---------- Binary, number system based on 2
BPS ---------- Bits per second
DAT ---------- Short for data appended to BASIC files
DC ---------- Direct Current
DEC ---------- Decimal, number system based on 10
DTACK ---------- Data Acknowledge signal on the MC68000
EPROM ---------- Erasable Read Only Memory
HC ---------- High speed Complementary Metal Oxide Silicon (CMOS)
HEX ---------- Hexadecimal, number system based on 16
I/O ---------- Input and Output
IC ---------- Integrated Circuit
K ---------- Thousand
KHZ ---------- Thousand Hertz (cycles) per second
LDS ---------- Lower Data Strobe signal for the MC68000
LS ---------- Low Power Shottky type of IC
LSB ---------- Least Significant Bit
MHZ ---------- Million Hertz (cycles) per second
MSB ---------- Most Significant Bit
OCT ---------- Octal, number system based on 8
PCM ---------- Pulse Code Modulation
R/W ---------- Read and Write signal on the MC68000
RAM ---------- Random Access Memory
RC ---------- Resistor and Capacitor network
ROM ---------- Read Only Memory
SWI ---------- Switch
TTL ---------- Transistor Transistor Logic, type of IC
UDS ---------- Upper Data Strobe signal for the MC68000

VITA

John R. Croasdale was born on 28 October 1944 in Ottumwa, Iowa. He graduated from Oviedo High School in Oviedo Florida in 1962 and attended the University of Florida where he received his Bachelor's of Science in Electrical Engineering in 1967 along with a commission in the United States Air Force. He attended pilot training at Moody AFB, Georgia and received his wings in 1968. His first assignment was in the Strategic Air Command flying KC-135 tanker aircraft at Warner Robins AFB, Ga. In 1972, he attended training at Castle AFB, California where he transitioned into the B-52 aircraft. He spent a total of 572 days flying in Southeast Asia in both tankers and bombers. In 1975 he was assigned to Wurtsmith AFB, Michigan as a B-52 Aircraft Commander and remained there until 1979 when he received a rated supplement job in engineering at Wright Patterson AFB, Ohio. He also received a Master of Arts degree in Industrial Management from Central Michigan University in 1979. When assigned to Wright Patterson AFB, he entered the School of Engineering, Air Force Institute of Technology as a part time student.

Permanent address: 405 Lake Drive

Chuluota, Florida 32766

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

AD-A164035

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS: None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for Public Release Distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GE/ENG/85S-1	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, OH 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: 4950th Test Wing; 8b. OFFICE SYMBOL (If applicable): 4950TESTW/FFS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code): Wright-Patterson AFB, OH 45433	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification): See Box 19	
12. PERSONAL AUTHOR(S): John R. Croasdale, Lt Col, USAF	
13a. TYPE OF REPORT: MS Thesis; 13b. TIME COVERED: FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day): 1985, September; 15. PAGE COUNT: 261
16. SUPPLEMENTARY NOTATION	

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	Pulse Code Modulation, Time Division Multiplexing, Data Acquisition, Microprocessor Applications
17			

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Title: Real-Time Flight Test PCM Data Acquisition Monitor

Thesis Chairman: David A. King, Captain, USAF
Instructor of Electrical Engineering

Approved for public release: IAW AFR 190-1.
16 JAN 86
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AFIT)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: David A. King, Captain, USAF	22b. TELEPHONE NUMBER (Include Area Code): 513-255-3576; 22c. OFFICE SYMBOL: AFIT/ENG

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE

unclassified

SECURITY CLASSIFICATION OF THIS PAGE

Block 19.

Abstract

A computer based system utilizing an inexpensive off-the-shelf personal computer and original interface design centered around a 68000 microprocessor for real-time monitoring of a time division multiplexed pulse code modulated (TDM/PCM) data stream was designed and constructed. This system is a prototype of a low-cost, portable PCM data acquisition monitor intended for use in flight test programs by the 4950th Test Wing at Wright-Patterson AFB, Oh. It will accept a single Armed Forces Instrumentation Standard NRZ or split-phase (Manchester) baseband data stream at rates up to 100 KBPS, display selected data words in graphical or numerical format, and alarm the user when data exceeds certain limits. It will provide a real-time verification that the data being generated and recorded during a test is of acceptable quality, allowing the option of continuation of the test, or termination. The system is capable of automatically determining the data rate and signaling format and synchronizing itself with the incoming signal.

END

FILMED

3 -86

DTIC